TU Clausthal
Clausthal University of Technology

Model Checking Abilities of Agents: A Closer Look

Wojciech Jamroga and Jürgen Dix

IfI Technical Report Series **IfI-06-02**

Department of Informatics
Clausthal University of Technology

Impressum

Publisher: Institut für Informatik, Technische Universität Clausthal
Julius-Albert Str. 4, 38678 Clausthal-Zellerfeld, Germany
Editor of the series: Jürgen Dix
Technical editor: Wojciech Jamroga
Contact: wjamroga@in.tu-clausthal.de
URL: http://www.in.tu-clausthal.de/forschung/technical-reports/
ISSN: 1860-8477

The IfI Review Board

Model Checking Abilities of Agents: A Closer Look

Wojciech Jamroga and Jürgen Dix

Department of Informatics, Clausthal University of Technology
Julius Albert Str. 4, D-38678 Clausthal Germany
{wjamroga,dix}@in.tu-clausthal.de

Abstract

Alternating-time temporal logic (ATL) is an extension of the branching-time temporal logic CTL, designed for reasoning about open computational systems and multi-agent systems. Like CTL, ATL has also been proven to enjoy model checking *linear in the size of the model*. We point out, however, that—unlike in CTL—the size of an ATL model is usually *exponential in the number of agents*. We establish the precise ATL model checking complexity for explicit models when the size of models is defined in terms of *states* rather than *transitions*: it turns out that the problem is $\mathbf{\Sigma_2^P}$-complete for concurrent game structures, and $\mathbf{NP}$-complete for alternating transition systems. We also discuss the determinism assumption in the semantics based on alternating transition systems, and show that this assumption can be easily removed.

We also show a *nondeterministic polynomial reduction* from model checking general *alternating transition systems* (ATS) to model checking *turn-based transition systems*. As model checking turn-based systems can be still done in deterministic polynomial time, the reduction provides an alternative proof that the original problem is in $\mathbf{NP}$. In our construction, the translation of models and formulae is done independently, which allows for "pre-compiling" models when one wants to check many properties of a particular system.

Finally, we study the complexity of model checking *alternating-time temporal logic with imperfect information*. We show that the problem is $\mathbf{NP}$-complete in the size of the model and the formula (thereby closing a gap in previous work of Schobbens [34]). Then, we take a closer look and use the same fine structure complexity measure as we did for ATL with perfect information. We get the surprising result that checking formulae of ATL with imperfect information is also $\mathbf{\Sigma_2^P}$-complete. Thus, both problems *belong to the same complexity class* when a finer-grained analysis is considered.

Keywords: multi-agent systems, model checking, computational complexity.

1 Introduction

Alternating-time temporal logic [3, 4, 5] is one of the most interesting frameworks that emerged recently for reasoning about computational systems. One of the most appreciated features of ATL is its model checking complexity—-*linear in the size of the model* (more precisely: the *number of transitions* in the model) *and the formula*. While the result is certainly attractive, it guarantees less than one could expect. We point out that the amount of transitions in an ATL model is usually exponential in the number of agents. While it is well-known that the number of states in a model can be exponential in the size of a higher-level description of the system, it also turns out that the size of an ATL model is usually *exponential in the number of agents,* even when no higher level description is considered.

Following this observation, we establish the precise ATL model checking complexity for *explicit models* when *the size of models is defined in terms of states* rather than transitions, and *the number of agents is considered a parameter of the problem*. In fact, we show that the model checking problem is intractable in such a setting of input parameters: it turns out that the problem is $\mathbf{\Sigma_2^P}$-complete for the ATL semantics based on concurrent game structures, and "only" $\mathbf{NP}$-complete when a previous semantics, based on ATS (alternating transition systems), is used. We also show that ATL model checking over the broader class of nondeterministic alternating transition systems is still $\mathbf{NP}$-complete, which suggests that using the more general class of models may be a good choice in practice.

On the other hand, model checking ATL for turn-based models (i.e. ones in which only one agent/process at a time is executing an action) can still be done in deterministic polynomial time. We show how, for an arbitrary alternating transition system M, a turn-based system M' can be constructed, so that a combination of *choices* in M corresponds to a combination of *strategies* in a fragment of M'. We then propose a translation of ATL formulae into ATL$^+$ formulae, such that the original formula holds in M, q iff the translated formula holds in M', q. Finally, we point out that the latter can be model-checked in nondeterministic polynomial time, and thus provide another (slightly more general) proof that the original problem is in NP. The translation of models is independent from the translation of formulae in our construction, which allows for "pre-compiling" models when one wants to check various properties of a particular multi-agent system.

The last part is concerned with ATL *with imperfect* (or *incomplete*) *information*. Since no satisfying semantics based on alternating transition systems has been proposed so far for strategic abilities under incomplete information, we present our results for an epistemic extension of concurrent game structures only. First, we close a gap in Schobbens's result and show that model checking an ATL formula with incomplete information is $\mathbf{NP}$-complete in the number of transitions and the length of the formula. Next, we demonstrate that model checking ATL with incomplete information is also $\mathbf{\Sigma_2^P}$-complete when the size of models is defined in terms of states rather than transitions. We point out that the result is somewhat surprising: *checking abilities of agents* *acting under incomplete information falls into the same complexity class as checking abilities of agents in perfect information scenarios* when a finer-grained analysis is conducted.

This article is organised as follows. In Section 2 we introduce ATL and its semantics, based on *concurrent game structures*. Several variants of ATL are considered and the notions of *perfect* and *imperfect* information in these systems are precisely defined. Section 3 presents classical results about the complexity of model checking (in particular the ATL model checking algorithm which we extend in later sections). Theorem 13 is our first result. In Section 4 we consider model checking with ATS and show that it is $\mathbf{NP}$-complete (Theorem 19). We also show that the usual *singleton* requirement in ATS can be relaxed without affecting the complexity. In Section 5 we relate ATS's and turn-based systems. Section 6 contains our main results: Theorems 26, 28 and Propositions 30, 31. They show, rather surprisingly, that there is no major difference in the complexity between games of perfect and imperfect information. We conclude with Section 7.

This article is based on preliminary results presented in [19, 20, 21].

2 ATL: A Logic of Strategic Ability

The logic of ATL [3, 4, 5] was originally invented to capture properties of *open computer systems* (such as computer networks), where different components can act autonomously, and computations in such systems are effected by their combined actions. Alternatively, ATL can be seen as a logic for systems involving multiple agents, that allows one to reason about what agents can achieve in game-like scenarios. ATL can be understood as a generalisation of the well-known branching time temporal logic CTL [13, 12], in which path quantifiers E ("there is a path") and A ("for every path") are replaced by *cooperation modalities* $\langle\langle A\rangle\rangle$ that express strategic abilities of agents and their teams.

Formula $\langle\langle A\rangle\rangle\varphi$ expresses that A have a collective strategy to enforce φ. ATL formulae include temporal operators: "$\bigcirc$" ("in the next state"), $\Box$ ("always from now on") and $\mathcal{U}$ ("until"). An additional operator $\Diamond$ ("sometime") can be defined as $\Diamond\varphi \equiv \top\,\mathcal{U}\,\varphi$. Like in CTL, every occurrence of a temporal operator is preceded by exactly one cooperation modality in ATL (this variant of the language is sometimes called "vanilla" ATL). The broader language of ATL*, in which no such restriction is imposed, is discussed briefly in Section 2.3.

Formally, the recursive definition of ATL formulae is:

$$\varphi := p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle A\rangle\rangle\bigcirc\varphi \mid \langle\langle A\rangle\rangle\Box\varphi \mid \langle\langle A\rangle\rangle\varphi\,\mathcal{U}\,\varphi$$

A number of different semantics and model classes have been defined for ATL, most of them equivalent (cf. [15, 16]). Among these, *concurrent game structures* [5] are probably the most natural and easiest to come up with when modelling concrete problem domains. Moreover, they are the easiest to extend to the incomplete information case, because actions have global identity in concurrent game structures (cf. [17]). However, it seems that *alternating transition systems*, introduced in the more preliminary papers [3, 4] may offer some advantage in terms of model checking complexity (see the results in Sections 3 and 4).

In what follows, we begin with a brief presentation of the two most prominent semantics, based on concurrent game structures and alternating transition systems. In Section 2.4, we are extending the scope of ATL with the possibility that some agents have incomplete information about the current state of the world. The research on this subject is far from being complete, yet a number of ATL extensions have already been proposed to cope with such systems: from the logics of ATEL [36, 37] and "ATL with incomplete information" [5] to more sophisticated approaches like ATOL and ATEL-R* [22], ATL_{ir} and ATL_{iR} [34], and ETSL [38]. Among these, ATL_{ir} seems to stand out for its simplicity and conceptual clarity; also (unlike "ATL with incomplete information", ATEL-R* and ATL_{iR}), its model checking procedure is decidable. We believe that ATL_{ir} – while probably *not* the definitive ATL extension for games incomplete information (ATOL, for example, is strictly more expressive with the same model checking complexity) – includes constructs that are indispensable when addressing such games. Thus, we treat ATL_{ir} as a kind of "core" ATL-based language for strategic ability under incomplete information, and present its syntax and semantics in Section 2.4.

2.1 Strategic Abilities with Concurrent Game Structures

Concurrent game structures (CGS) [5], can be defined as tuples

$$M = \langle \mathbb{A}\text{gt}, St, \Pi, \pi, Act, d, o \rangle,$$

where:

- $\mathbb{A}\text{gt} = \{a_1, ..., a_k\}$ is a finite nonempty set of all agents,
- St is a nonempty set of states,
- Π is a set of atomic propositions,
- $\pi : \Pi \to \mathcal{P}(St)$ is a valuation of propositions,
- Act is a finite nonempty set of (atomic) actions;
- function $d : \mathbb{A}\text{gt} \times St \to \mathcal{P}(Act)$ defines actions available to an agent in a state, and
- o is a (deterministic) transition function that assigns outcome states $q' = o(q, \alpha_1, \ldots, \alpha_k)$ to states and tuples of actions.

Remark 1 *Firstly, this variant of concurrent game structures differs slightly from the original* CGS *[5]: we represent agents and their actions with symbolic labels, whereas they are represented with natural numbers in [5].*

Secondly, determinism is not a crucial issue here, as systems with nondeterministic outcome of agents' actions can be modelled easily by introducing a new, fictional agent, "Nature", which settles all nondeterministic transitions.

A *strategy* of agent a is a conditional plan that specifies what a is going to do in every possible situation (state). Thus, a strategy can be represented with a function $s_a : St \rightarrow Act$, such that $s_a(q) \in d_a(q)$. A *collective strategy* for a group of agents $A = \{a_1, ..., a_r\}$ is simply a tuple of strategies $S_A = \langle s_{a_1}, ..., s_{a_r}\rangle$, one per agent from A.

Remark 2 *This is an important deviation from the original semantics of* ATL *[3, 4, 5], where strategies assign agents' choices to* sequences *of states, which suggests that agents can recall the whole history of each game. In this article, however, we employ "memoryless" strategies. While the choice of one or another notion of strategy affects the semantics of the full* ATL**, and most* ATL *extensions (e.g. for games with incomplete information), it should be pointed out that both types of strategies yield equivalent semantics for "pure"* ATL *[34].*

A *path* in M is an infinite sequence of states that can be affected by subsequent transitions, and refers to a possible course of action (or a possible computation). Function $out(q, S_A)$ returns the set of all paths that may result from agents A executing strategy S_A from state q onward.

$out(q, S_A) = \{\lambda = q_0 q_1 q_2 ... \mid q_0 = q$ and for every $i = 1, 2, ...$ there exists a tuple of agents' decisions $\langle \alpha^{i-1}_{a_1}, ..., \alpha^{i-1}_{a_k}\rangle$ such that $\alpha^{i-1}_a \in d_a(q_{i-1})$ for each $a \in A$, and $o(q_{i-1}, \alpha^{i-1}_{a_1}, ..., \alpha^{i-1}_{a_k}) = q_i\}$.

Let $\Lambda[i]$ denote the ith position in computation Λ (starting from $i = 0$). The semantics of ATL is defined via the clauses below. Informally speaking, $M, q \models \langle\langle A\rangle\rangle\Phi$ iff there exists a collective strategy S_A such that Φ holds for all computations from $out(q, S_A)$.

$M, q \models p$ iff $q \in \pi(p)$ (where $p \in \Pi$);

$M, q \models \neg\varphi$ iff $M, q \not\models \varphi$;

$M, q \models \varphi \vee \psi$ iff $M, q \models \varphi$ or $M, q \models \psi$;

$M, q \models \langle\langle A\rangle\rangle \bigcirc \varphi$ iff there is a collective strategy S_A such that, for every path $\lambda \in out(S_A, q)$, we have $M, \lambda[1] \models \varphi$;

$M, q \models \langle\langle A\rangle\rangle \Box \varphi$ iff there exists S_A such that, for every $\lambda \in out(S_A, q)$, we have $M, \lambda[i]$ for every $i \geq 0$;

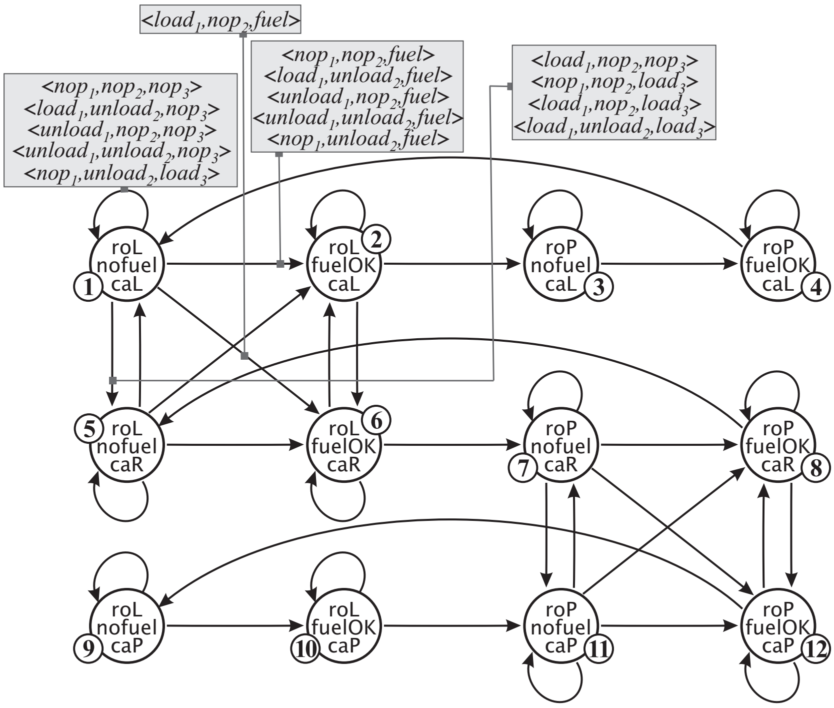


Figure 1: A CGS for Simple Rocket Domain

$M, q \models \langle\langle A \rangle\rangle \varphi \mathcal{U} \psi$ iff there exists S_A such that, for every $\lambda \in out(S_A, q)$, there is $i \geq 0$ for which $M, \lambda[i] \models \psi$, and $M, \lambda[j] \models \varphi$ for every $0 \leq j < i$.

Example 1 *As an example, consider a modified version of the Simple Rocket Domain from [7]. There is a rocket that can be moved between London (*roL*) and Paris (*roP*), and piece of cargo that can lie in London (*caL*), Paris (*caP*), or inside the rocket (*caR*). Three agents are involved:* 1 *who can load the cargo, unload it, or move the rocket;* 2 *who can unload the cargo or move the rocket, and* 3 *who can load the cargo or supply the rocket with fuel. Every agent can also stay idle at a particular moment (the* nop *– "no-operation" actions). The "moving" action has the highest priority. "Loading" is affected when the rocket does not move and more agents try to load than to unload; "unloading" works in a similar way (in a sense, the agents "vote" whether the cargo should be loaded or unloaded). Finally, "fuelling" can be accomplished only when the rocket tank is empty (alone or in parallel with loading or unloading). The rocket can move only if it has some fuel (*fuelOK*), and the fuel must be refilled after each flight. The concurrent game structure for the domain is shown in Figure 1 (we will refer to this* *model as M_1). All the transitions for state 1 (the cargo and the rocket are in London, no fuel in the rocket) are labelled; output of agents' choices for other states is analogous.*

Example ATL *formulae that hold in $M_1, 1$ are: $\neg\langle\langle 1\rangle\rangle\Diamond\mathsf{caP}$ (agent 1 cannot deliver the cargo to Paris on his own), $\langle\langle 1, 3\rangle\rangle\Diamond\mathsf{caP}$ (1 and 3 can deliver the cargo if they cooperate), and $\langle\langle 2, 3\rangle\rangle\Box(\mathsf{roL} \wedge \langle\langle 2, 3\rangle\rangle\Diamond\mathsf{roP})$ (2 and 3 can keep the rocket in London forever, and at the same time retain the ability to change their strategy and move the rocket to Paris).*

It is worth pointing out that the CTL path quantifiers A and E can be embedded in ATL in the following way: $\mathsf{A}\varphi \equiv \langle\langle\varnothing\rangle\rangle\varphi$ and $\mathsf{E}\varphi \equiv \langle\langle\mathbb{A}\mathrm{gt}\rangle\rangle\varphi$. Note that the determinism of the transition function is essential for the latter property. In a deterministic system, a collective strategy for the "grand coalition" of agents $\mathbb{A}\mathrm{gt}$ determines a *single* path in the model. In contrast, this is usually not the case in non-deterministic systems. Thus, it may be the case that there is a single path for which property φ holds (i.e., we have $\mathsf{E}\varphi$), and yet the agents are not able to enforce it, so $\langle\langle\mathbb{A}\mathrm{gt}\rangle\rangle\varphi$ does not hold (see also Remark 4).

On the other hand, $\mathsf{A}\varphi$ is still equivalent to $\langle\langle\varnothing\rangle\rangle\varphi$ even when we abandon the determinism assumption (to see this, it is sufficient to check what the semantic clauses for $\langle\langle\varnothing\rangle\rangle\bigcirc\varphi$, $\langle\langle\varnothing\rangle\rangle\Box\varphi$ and $\langle\langle\varnothing\rangle\rangle\varphi\,\mathcal{U}\,\psi$ look like).

2.2 Semantics of ATL Based on ATS

Previous versions of ATL were defined over alternating transition systems [3, 4]. An *alternating transition system* (ATS) is a tuple

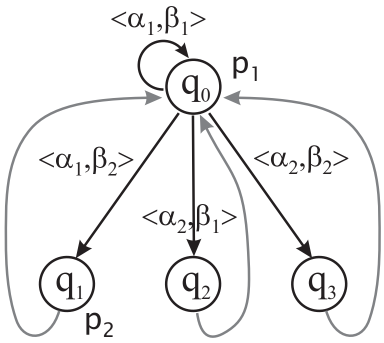
$$M = \langle \mathbb{A}\mathrm{gt}, St, \Pi, \pi, \delta\rangle,$$

where:

- $\mathbb{A}\mathrm{gt}$ is a non-empty finite set of *agents*, St is a non-empty set of *states*, Π is a set of (atomic) *propositions*, and $\pi : St \to \mathcal{P}(\Pi)$ is a *valuation* of propositions;

- $\delta : St \times \mathbb{A}\mathrm{gt} \to \mathcal{P}(\mathcal{P}(St))$ is a function that maps pairs $\langle state, agent\rangle$ to non-empty families of choices with respect to possible next states. The idea is that, at state q, agent a chooses a set $Q_a \in \delta(q, a)$ thus forcing the outcome state to be from Q_a. The resulting transition leads to a state which is in the intersection of all Q_a for $a \in \mathbb{A}\mathrm{gt}$ and so it reflects the will of all agents. Since the system is required to be deterministic (given the state and the agents' decisions), $Q_{a_1} \cap \ldots \cap Q_{a_k}$ must always be a singleton.

In an ATS, the type of a strategy function is slightly different since choices are sets of states now, and a strategy is represented as a mapping $s_a : St \to \mathcal{P}(St)$, such that $s_a(q) \in \delta(q, a)$. The rest of the semantics looks exactly the same as for concurrent game structures. In particular, the semantic clauses are exactly the same as the ones in Section 2.1.



$$\mathbb{A}\mathrm{gt} = \{a, b\}$$
$$St = \{q_0, q_1, q_2, q_3\}$$
$$\Pi = \{p_1, p_2\}$$
$$\pi(p_1) = \{q_0\},\ \pi(p_2) = \{q_1\}$$

$$\delta(q_0, a) = \{\overset{\alpha_1}{\{q_0, q_1\}}, \overset{\alpha_2}{\{q_2, q_3\}}\}$$
$$\delta(q_0, b) = \{\overset{\beta_1}{\{q_0, q_2\}}, \overset{\beta_2}{\{q_1, q_3\}}\}$$

$$\delta(q_1, b) = \delta(q_2, a) = \delta(q_3, a) = \{\{q_0\}\}$$
$$\delta(q_1, a) = \delta(q_2, b) = \delta(q_3, b) = \{\{q_0, q_1\}\}$$

Figure 2: Alternating transition system M_2: 2 agents, each has two choices at state q_0

Example 2 *Consider* ATS M_2 *from Figure 2. We use symbols α_1, α_2 and β_1, β_2 as shorthand for the choices, to make the example easier to read. The following example* ATL *formulae hold in M_2, q_0: $\neg\langle\langle a\rangle\rangle\Diamond \mathsf{p}_1$ (a cannot enforce that p_1 is eventually true), $\langle\langle a, b\rangle\rangle\Box \mathsf{p}_1$ (a and b can cooperate to guarantee that p_1 always holds), and $\langle\langle a\rangle\rangle\bigcirc(\mathsf{p}_1 \vee \mathsf{p}_2)$ (a can achieve $\mathsf{p}_1 \vee \mathsf{p}_2$ in the next step).*

Note that M_2 is not "tight" in the sense that some choices include states that cannot be reached via these choices. It can be tightened by removing q_1 from the choices in $\delta(q_1, a)$, $\delta(q_2, b)$ and $\delta(q_3, b)$, which yields an equivalent tight ATS*. We discuss the notion of tightness in Section 4.1 more formally.*

It is worth pointing out that alternating transition systems are usually less natural and more difficult to come up with than concurrent game structures; they are also larger in most cases (cf. [18], Section 2.7.4). More precisely: for every ATS there exists an isomorphic CGS, but the reverse does not hold. Moreover, alternating transition systems do not lend themselves easily to extensions (e.g. with the possibility that agents may have incomplete information). This subject was discussed in more detail in [17, 16, 18].

Remark 3 *Note that the determinism assumption* is *significant in the case of* ATS*. Unlike for* CGS*, adding an auxiliary player ("Nature") to an existing alternating transition system is neither easy nor straightforward. The problem is to extend the existing choice function δ so that it still satisfies the rigid formal requirement that* all *the intersections of choices are singletons. Designing a completely new* ATS *from scratch is probably an easier solution.*

We note here that model checking of ATL formulae has been proven linear in the size of the model and the length of the formula for both concurrent game structures [5] and alternating transition systems [4], which coincides with the model checking complexity for CTL [8]. We will discuss this issue in more detail in Section 3.1.

2.3 Beyond ATL: ATL$^+$ and ATL*

The full language of ATL* [4, 5] is usually presented as consisting of *state formulae* $\langle\langle A\rangle\rangle\varphi$ (expressing strategic abilities of agents to enforce specific paths of computation) and *path formulae* $\bigcirc\varphi$ and $\varphi\,\mathcal{U}\,\psi$ (expressing temporal properties of paths). Both state and path formulae can be combined using Boolean operators. State formulae are interpreted in states, with $M, q \models \langle\langle A\rangle\rangle\varphi$ meaning "there is S_A such that, for every path $\Lambda \in out(q, S_A)$, we have $M, \Lambda \models \varphi$". Path formulae are interpreted in paths, with $M, \Lambda \models \bigcirc\varphi$ and $M, \Lambda \models \varphi\,\mathcal{U}\,\psi$ defined in the obvious way. ATL* is more costly in computational terms. Model checking ATL* with memoryless strategies (i.e., the variant that we are interested in here) is **PSPACE**-complete [34]. Model checking ATL* with perfect recall is even more expensive: it is 2EXPTIME-complete in the number of transitions in the model and the length of the formula [5].

In this article, we are only interested in its subset ATL$^+$ [34], in which every temporal operator is preceded by a single cooperation modality, modulo Boolean operators. That is, $\langle\langle A\rangle\rangle$ is followed by a Boolean combination of path formulae $\bigcirc\varphi$, $\varphi\,\mathcal{U}\,\psi$, in which φ, ψ are state formulae again. As an example, the following is an ATL$^+$ formula: $\langle\langle a\rangle\rangle(\Box\Diamond \mathsf{p}_2 \wedge \Box\Diamond\neg(\mathsf{p}_1 \vee \mathsf{p}_2))$. It states that a has a strategy to visit state q_1 and at least one of states q_2, q_3 infinitely often (note, by the way, that the formula holds in M_2, q_0 from Example 2).

ATL$^+$ can be seen as a generalisation of CTL$^+$ [14]. Model checking of ATL$^+$ has been proved Δ_3-complete in the number of transitions and the length of the formula (for both memoryless and perfect recall strategies) [34], while CTL$^+$ model checking is Δ_2-complete [27]. However, the ATL$^+$ and CTL$^+$ formulae that we use in this article can be model checked in nondeterministic polynomial time (cf. Section 5.2).

2.4 Strategic Abilities under Incomplete Information

ATL and its models include no way of addressing uncertainty that an agent or a process may have about the current situation; moreover, strategies in ATL can define different choices for any pair of different states, hence implying that an agent can recognise each (global) state of the system, and act accordingly. Thus, it can be argued that the logic is tailored for describing and analyzing systems in which every agent/process has *complete and accurate knowledge* about the current state of the system. This is usually not the case for most application domains, where a process can access its *local* state, but the state of the environment and the (local) states of other agents can be observed only partially.

One of the main challenges, when a logic of strategic abilities under incomplete information is addressed, is the question of how agents' knowledge should interfere with the agents' available strategies. When reasoning about what an agent can *enforce*, it seems more appropriate to require the agent to know his winning strategy rather than to know only that such a strategy exists [17, 22, 23]. This problem is closely related to the distinction between knowledge *de re* and knowledge *de dicto*, well known in the philosophy of language [33], as well as in research on the interaction between knowledge and action [30, 31, 39]. Several variations on "ATL with incomplete information" have been proposed [22, 34, 23, 38], yet none of them seems the ultimate definitive solution. In this article, we treat Schobbens' ATL_{ir} and ATL_{iR} [34] as "core", minimal ATL-based languages for strategic ability under incomplete information. The first logic enables reasoning about agents that have no implicit memory of the game (i.e., they use "memoryless" strategies), while the latter is underlain by the assumption that agents can always memorise the whole game. As agents seldom have unlimited memory, and logics of strategic ability with incomplete information and perfect recall are believed to have undecidable model checking, we use ATL_{ir} as *the* logic of strategic ability under uncertainty here.

ATL_{ir} includes the same formulae as ATL, only the cooperation modalities are presented with a subscript: $\langle\langle A\rangle\rangle_{ir}$ to indicate that they address agents with imperfect *information* and imperfect *recall*. Models of ATL_{ir}, *imperfect information concurrent game structures* (i-CGS), can be presented as concurrent game structures augmented with a family of indistinguishability relations $\sim_a \subseteq St \times St$, one per agent $a \in \mathbb{A}\text{gt}$. The relations describe agents' uncertainty: $q \sim_a q'$ means that, while the system is in state q, agent a considers it possible that it is in q' now. Every $\sim_a$ is assumed to be an equivalence. It is required that agents have the same choices in indistinguishable states: if $q \sim_a q'$ then $d(a,q) = d(a,q')$.

Again, a *strategy* of agent a is a conditional plan that specifies what a is going to do in every possible state. An executable (deterministic) plan must prescribe the same choices for indistinguishable states. Therefore ATL_{ir} restricts the strategies that can be used by agents to the set of so called uniform strategies. A *uniform strategy* of agent a is defined as a function $s_a : St \to Act$, such that: (1) $s_a(q) \in d(a,q)$, and (2) if $q \sim_a q'$ then $s_a(q) = s_a(q')$. A *collective strategy* for a group of agents $A = \{a_1, ..., a_r\}$ is a tuple of strategies $S_A = \langle s_{a_1}, ..., s_{a_r}\rangle$, one per each agent from A. A collective strategy is uniform if it contains only uniform individual strategies. Again, function $out(q, S_A)$ returns the set of all paths that may result from agents A executing strategy S_A from state q onward. The semantics of cooperation modalities in ATL_{ir} is defined as follows:

$M, q \models \langle\langle A\rangle\rangle_{ir} \bigcirc \varphi$ iff there is a uniform collective strategy S_A such that, for every $a \in A$, q' such that $q \sim_a q'$, and path $\lambda \in out(S_A, q')$, we have $M, \lambda[1] \models \varphi$;

$M, q \models \langle\langle A\rangle\rangle_{ir} \Box \varphi$ iff there exists a uniform S_A such that, for every $a \in A$, q' such that $q \sim_a q'$, and $\lambda \in out(S_A, q')$, we have $M, \lambda[i]$ for every $i \geq 0$;

$M, q \models \langle\langle A\rangle\rangle_{ir} \varphi \mathcal{U} \psi$ iff there exist a uniform strategy S_A such that, for every $a \in A$, q' such that $q \sim_a q'$, and $\lambda \in out(S_A, q')$, there is $i \geq 0$ for which $M, \lambda[i] \models \psi$, and $M, \lambda[j] \models \varphi$ for every $0 \leq j < i$.

That is, $\langle\langle A\rangle\rangle_{ir}\varphi$ if A have a uniform strategy, such that for every path *that can possibly result from execution of the strategy according to at least one agent from* A, φ is the case.

Schobbens [34] proved that ATL_{ir} model checking is $\mathbf{NP}$-hard and $\mathbf{\Delta_2^P}$-easy. He also conjectured that the problem is probably $\mathbf{\Delta_2^P}$-complete. We discuss the issue in more detail, and provide a more definitive result in Section 6.

Remark 4 *The* CTL *universal path quantifier* A *can be expressed in* ATL_{ir} *in the following way:* $\mathsf{A}\varphi \equiv \langle\langle\varnothing\rangle\rangle_{ir}\varphi$*. The existential path quantifier* E*, however, is not fully expressible when cooperation modalities quantify over uniform strategies only. Like for non-deterministic models, it may be the case that there is a single path for which property* φ *holds (i.e., we have* $\mathsf{E}\varphi$*), and yet even the "grand coalition" of agents* $\mathbb{A}\text{gt}$ *is not able to enforce it, so* $\langle\langle\mathbb{A}\text{gt}\rangle\rangle_{ir}\varphi$ *does not hold. Moreover,* $\mathsf{E}\varphi\,\mathcal{U}\,\psi$ *cannot be expressed as a combination of* $\mathsf{A}\varphi\,\mathcal{U}\,\psi$, $\mathsf{E}\Diamond\varphi$, $\mathsf{E}\Box\varphi$, $\mathsf{A}\Box\varphi$, $\mathsf{E}\bigcirc\varphi$, *and* $\mathsf{A}\bigcirc\varphi$ *(cf. [26]).*

3 Complexity of ATL Model Checking Revisited

The model checking problem asks, given model M, state q in M, and formula φ, whether φ holds in M, q. Model checking of temporal logics is usually computationally cheaper than satisfiability checking or theorem proving, while often being at least as useful because the designer or user of a system can come up with a precise model of the system behaviour (e.g. a graph with all the actions that may be affected) in many cases. For ATL, model checking has been proved *linear in the size of the models and formulae*. This seems to be a very good property, but unfortunately it guarantees less than one could expect.

3.1 Model Checking ATL: Easy or Hard?

It has been known for a long time that formulae of CTL can be checked in time linear with respect to the size of the model and the length of the formula [8]. One of the main results concerning ATL states that its formulae can also be model-checked in deterministic linear time.

Proposition 5 [4, 5] *The* ATL *model checking problem is* PTIME*-complete, and can be done in time* $\mathbf{O}(ml)$*, where* m *is the number of transitions in the model and* l *is the length of the formula.*

The ATL *model checking algorithm from [4, 5] is presented in Figure 3.*

While the result is certainly attractive, it should be kept in mind that it is only relative to the size of models and formulae, and these can be very large for most application domains. Indeed, it is well known that the number of states in a model is usually exponential in the size of a higher-level description of the problem domain for both CTL and ATL models. Consider, for example, a system whose state space is defined through r Boolean variables (binary attributes). Obviously, the number of global states in the system is then $n = 2^r$. A more general approach is presented in [25], where the "higher level description" is defined in terms of so called *concurrent programs*, that can

function $mcheck_1(M, \varphi)$.
Returns the set of states in model $M = \langle \mathbb{A}\mathrm{gt}, St, \Pi, \pi, o \rangle$ for which formula φ holds.
case $\varphi \in \Pi$ **:** return $\pi(p)$ **case** $\varphi = \neg\psi$ **:** return $St \setminus mcheck_1(M, \psi)$ **case** $\varphi = \psi_1 \vee \psi_2$ **:** return $mcheck_1(M, \psi_1) \cup mcheck_1(M, \psi_2)$ **case** $\varphi = \langle\langle A \rangle\rangle \bigcirc \psi$ **:** return $pre_1(M, A, mcheck_1(M, \psi))$ **case** $\varphi = \langle\langle A \rangle\rangle \Box \psi$ **:** $Q_1 := St$; $Q_2 := mcheck_1(M, \psi)$; $Q_3 := Q_2$; **while** $Q_1 \not\subseteq Q_2$ **do** $Q_1 := Q_2$; $Q_2 := pre_1(M, A, Q_1) \cap Q_3$ **od**; return Q_1 **case** $\varphi = \langle\langle A \rangle\rangle \psi_1 \,\mathcal{U}\, \psi_2$ **:** $Q_1 := \emptyset$; $Q_2 := mcheck_1(M, \psi_1)$; $Q_3 := mcheck_1(M, \psi_2)$; **while** $Q_3 \not\subseteq Q_1$ **do** $Q_1 := Q_1 \cup Q_3$; $Q_3 := pre_1(M, A, Q_1) \cap Q_2$ **od**; return Q_1 **end case**

function $pre_1(M, A, Q)$.
Auxiliary function, returns the exact set of states Q' such that, when the system is in a state $q \in Q'$, agents A can cooperate and enforce the next state to be in Q.
return $\{q \mid \exists\alpha_A \forall\alpha_{\mathbb{A}\mathrm{gt}\setminus A}\ o(q, \alpha_A, \alpha_{\mathbb{A}\mathrm{gt}\setminus A}) \in Q\}$

Figure 3: The ATL model checking algorithm from [5]

be used for simulating Boolean variables, but also processes or agents acting in parallel. Each concurrent program $C = \langle C_1, ..., C_k \rangle$ implicitly generates a system of global states which is defined as the product automaton of C. The main result concerning model checking is that checking CTL formulae is **PSPACE**-complete in the size of the concurrent program (and the length of the formula) [25].[1]

Thus, there are basically two kinds of results regarding model checking CTL and ATL. On the one hand, the problem is computationally easy with respect to CTL/ATL models one uses when defining semantics (sometimes called *global state graphs* [8] or *explicit models* [29]). On the other hand, the problem is very hard with respect to more compact representations (e.g. concurrent programs), mainly because these representations unravel into exponentially large explicit models. As a concurrent program may be

[1] We also note in passing that, for *some* high-level system descriptions, even the computation of $\langle\langle A \rangle\rangle \bigcirc$ may require **PSPACE** or even **NEXPTIME** [9, 10], but these results are not relevant for our discussion here.

seen as a system involving k agents, this already shows that having multiple agents can make models (and model checking) explode *with respect to a high level description*. What we point out in this article is that the complexity of $\mathbf{O}(ml)$ includes potential intractability *even on the level of explicit models* if the size of models is defined in terms of states rather than transitions, and the number of agents is a parameter of the problem rather than a fixed value. We state the observation formally as follows.

Remark 6 *Let n be the number of states in an* ATL *model M. It was already observed in [5] that the number of transitions in M is* not *bounded by n^2, because transitions are labelled with tuples of agents' choices. Here, we make the observation more precise.*

Let k denote the number of agents, and d the maximal number of available decisions per agent per state. Then, $m = O(nd^k)$. In consequence, the ATL *model checking algorithms from [4, 5] run in time $O(nd^k l)$, and hence their complexity is exponential if the number of agents is a parameter of the problem.*

Example 1 is quite illustrative in this respect. The state space refers to valuations of only three attributes (two binary, and one ternary), which yields 12 states. And the number of transitions is already 216, despite the fact that the system includes only three agents, and every agent has only two or three actions available at each state.

Remark 7 *Note that, for turn-based* ATS*, only one agent is playing at a time, so the number of transitions is $O(nd)$, and hence model checking can be done in time $O(ndl)$.*

Throughout the rest of Section 3, we establish the complexity of model checking ATL formulae over concurrent game structures, with n, k, d, l as input parameters. We show that the problem is $\mathbf{\Sigma_2^P}$-complete, where $\mathbf{\Sigma_2^P} = \mathbf{NP^{NP}}$ is the class of problems that can be solved by a nondeterministic Turing machine in polynomial time with calls to an $\mathbf{NP}$ oracle. The result seems natural as soon as we re-formulate $M, q \models \langle\langle a_1, ..., a_r \rangle\rangle \bigcirc \varphi$ as $\exists(\alpha_1, ..., \alpha_r)\forall(\alpha_{r+1}, ..., \alpha_k)\ M, o(q, \alpha_1, ..., \alpha_k) \models \varphi$, which bears close resemblance to the problem of QSAT_2. Before we prove it formally, however, we must make one more important remark.

Remark 8 *Note that the transition function o must be kept externally to the model checking algorithm, or represented in a somehow "compressed" way. Otherwise the algorithm requires exponential amount of memory to store the function, and in consequence the problem is not even in* PSPACE*. In what follows, we assume that the transition function can be implemented as an external procedure (more precisely: deterministic Turing Machine) that, given state q and actions $\alpha_1, ..., \alpha_k$, returns the value of $o(q, \alpha_1, ..., \alpha_k)$ in polynomial time.*

3.2 ATL Model Checking for Concurrent Game Structures is $\mathbf{\Sigma_2^P}$-hard

First, we show that model checking of ATL formulae over concurrent game structures is $\mathbf{\Sigma_2^P}$-hard. We show this through a polynomial reduction of QSAT_2 to the model checking problem. In QSAT$_i$ (satisfiability for quantified Boolean formulae with i alternations of quantifiers), we are given k propositional variables $p_1, ..., p_k$ (partitioned into i sets $P_1, ..., P_i$) and a Boolean formula θ that includes no other variables. QSAT$_i$ asks if $\exists P_1 \forall P_2 \exists P_3 \ldots \Delta P_i\ \theta$ (where $\Delta = \forall$ if i is even, and $\exists$ if i is odd), i.e. whether there is a valuation of propositions in P_1 such that, for all valuations of propositions in P_2, there exists a valuation of propositions in P_3 etc., such that θ is satisfied. (We will use θ as a symbol for the Boolean formula that appears in the QSAT problem, to distinguish it from the ATL formula in the model checking problem which we usually denote with φ). QSAT$_i$ is known to be $\mathbf{\Sigma_i^P}$-complete [32].

To obtain the reduction, we construct a concurrent game structure M with 3 states: $St = \{q_0, q_\top, q_\bot\}$, and k agents: $\mathbb{A}\mathrm{gt} = \{a_1, ..., a_k\}$ that "decide" at q_0 upon valuations of propositions $p_1, ..., p_k \in P_1 \cup P_2$, respectively. Thus, agent a_i can "declare" proposition p_i true (action $\top$) or false (action $\bot$); Every tuple of actions from $\mathbb{A}\mathrm{gt}$ corresponds to a valuation $v_1, ..., v_k$ of the propositions, and vice versa. Now, the transitions from q_0 are defined in the following way:

$$o(q_0, v_1, ..., v_k) = \begin{cases} q_\top & \text{if } v_1, ..., v_k \models \theta \\ q_\bot & \text{else} \end{cases}$$

Transitions from $q_\top$ and $q_\bot$ do not matter. Note that $v_1, ..., v_k \models \theta$ can be verified in time and space linear in $|\theta|$, so o has a polynomial representation with respect to the size of the original problem. Finally, we define proposition sat to hold only in state $q_\top$. Note that the agents "controlling" propositions from P_1 can enforce the next state to be $q_\top$ if, and only if, they can declare such a valuation of "their" propositions that θ is satisfied regardless of the opponents' choices:

Lemma 9 *Let A be the group of agents "responsible" for propositions P_1, i.e. $a_i \in A$ iff $p_i \in P_1$. Then, $\exists P_1 \forall P_2\ \theta$ iff $M, q_0 \models \langle\!\langle A \rangle\!\rangle \bigcirc \mathsf{sat}$.*

Proposition 10 *Model checking formulae of* ATL *over concurrent game structures is $\mathbf{\Sigma_2^P}$-hard.*

3.3 ATL Model Checking for Concurrent Game Structures is $\mathbf{\Sigma_2^P}$-easy

In order to demonstrate $\mathbf{\Sigma_2^P}$-easiness of the model checking problem, we show an algorithm that computes the set of states in which formula φ holds, and lies in $\mathbf{NP^{NP}}$. A careful analysis of the algorithms proposed in [4, 5] reveals that the intractability is due to the pre-image operator pre, which is called at most n times for every subformula of φ. Indeed, as we saw in the previous section, checking what a coalition can enforce in a single step (e.g., $M, q \models \langle\!\langle A \rangle\!\rangle \bigcirc \mathsf{sat}$) lies very close to the standard $\mathbf{\Sigma_2^P}$-complete problem of QSAT$_2$. We show that checking a more sophisticated ATL formula is no more

function $mcheck_2(M, \varphi)$**;**
Returns the set of states in M, in which formula φ holds.

- assign cooperation modalities in φ with subsequent numbers $1, ..., c$;
  `// note that` $c \leq l$; $c(\varphi)$ `is the number of cooperation modalities in` φ
  `// we denote the coalition from the` i`th coop. modality in` φ `as` $\varphi[i]$
- for every $i = 1, ..., c$, assign the agents in $\varphi[i]$ with numbers $1, ..., k_c$;
  `// note that` $k_c \leq k$ `and` $k_c \leq l$
  `// we will denote the` j`th agent in` A `with` $A[j]$
- guess an array $choice$ such that, for every $i = 1, ..., c$, $q \in St$, and $j = 1, ..., k_c$, we have that $choice[i][q][j] \in d_{\varphi[i][j]}(q)$;
  `// now, the optimal choices for all coalitions in` φ `are guessed`
  `// note that the size of` $choice$ `is` $O(nkl)$
  `// by` $choice|_i$`, we denote array` $choice$ `with rows` $1, ..., i-1$ `removed`
- return $eval_2(M, \varphi, choice)$;

function $eval_2(M, \varphi, choice)$**;**
Returns the states in which φ holds, given choices for all the coalitions from φ.

case $\varphi \in \Pi$ **:** return $\{q \mid \varphi \in \pi(q)\}$;
case $\varphi = \neg\psi$ **:** return $Q \setminus eval_2(M, \psi, choice)$;
case $\varphi = \psi_1 \vee \psi_2$ **:** return $eval_2(M, \psi_1, choice) \cup eval_2(M, \psi_2, choice|_{c(\psi_1)+1})$;
case $\varphi = \langle\langle A \rangle\rangle \bigcirc \psi$ **:** return $pre_2(M, A, eval_2(M, \psi, choice|_2), choice[1])$;
case $\varphi = \langle\langle A \rangle\rangle \Box \psi$ **:** $Q_1 := St$; $Q_2 := Q_3 := eval_2(M, \psi, choice|_2)$;
  while $Q_1 \not\subseteq Q_2$ **do** $Q_1 := Q_1 \cap Q_2$; $Q_2 := pre_2(M, A, Q_1, choice[1]) \cap Q_3$ **od**;
  return Q_1;
case $\varphi = \langle\langle A \rangle\rangle \psi_1 \mathcal{U} \psi_2$ **:** $Q_1 := \varnothing$; $Q_2 := eval_2(M, \psi_1, choice|_2)$;
  $Q_3 := eval_2(M, \psi_2, choice|_{c(\psi_1)+2})$;
  while $Q_3 \not\subseteq Q_1$ **do** $Q_1 := Q_1 \cup Q_3$; $Q_3 := pre_2(M, A, Q_1, choice[1]) \cap Q_2$ **od**;
  return Q_1;
end case

Figure 4: Nondeterministic algorithm for model checking formulae of ATL: part I.

complex than this. The main idea of the algorithm is as follows. First, we guess non-deterministically *all* the choices that will be needed for any call to function Pre (that is, for each coalition A that occurs in φ, and for each state $q \in St$). Then we employ the standard model checking algorithm from Figure 3 with one important modification: every time function $pre_2(M, A, Q_1)$ is called, it assumes the subsequent A's choices

function $pre_2(M, A, Q_1, thischoice)$**;**
Returns the set of states, for which the A's choices from $thischoice$ enforce that the next state is in Q_1, regardless of what agents from $\mathbb{A}\mathrm{gt} \setminus A$ do.

- $Q_2 := \varnothing$;
- for each $q \in St$: **if** $oracle_2(M, A, Q_1, thischoice, q) = yes$ **then** $Q_2 := Q_2 \cup \{q\}$ **fi**;
- return Q_2;

function $oracle_2(M, A, Q_1, thischoice, q)$**;**
Returns yes if, and only if, the A's choices from $thischoice$ in q enforce that the next state is in Q_1, regardless of what agents from $\mathbb{A}\mathrm{gt} \setminus A$ do.

- guess an array $resp$ such that, for every $a \in \mathbb{A}\mathrm{gt} \setminus A$, we have $resp[a] \in d_a(q)$;
  `// the most dangerous response from the opposition is guessed`
  `// note that the size of` $resp$ `is` $O(k)$
- **if** $o(q, thischoice[q], resp) \in Q_1$ **then** return yes **else** return no **fi**;

Figure 5: Nondeterministic algorithm for model checking formulae of ATL: part II.

from the tuple and checks whether $q \in pre_2(M, A, Q_1)$ by calling an $\mathbf{NP}$ oracle (*is there a response from the opposition in* q *that leads to a state outside* Q_1*?*) and invert its answer. The detailed algorithm is shown in Figures 4 and 5. As the number of iterations, as well as the number of calls to pre, in the algorithm from Figure 3 is $O(nl)$, we get a nondeterministic polynomial algorithm that makes calls to an $\mathbf{NP}$ oracle.

Lemma 11 *Function* $mcheck_2$ *defines a nondeterministic Turing machine that runs in time* $O(nkl)$*, making calls to an* $\mathbf{NP}$ *oracle. The size of the witness is* $O(nkl)$*. The oracle is a nondeterministic Turing machine that runs in time* $O(n + k)$*.*

Proposition 12 *Model checking formulae of* ATL *over concurrent game structures is* $\mathbf{\Sigma_2^P}$*-easy.*

The following theorem is an immediate corollary:

Theorem 13 *Model checking* ATL *formulae over* CGS *is* $\mathbf{\Sigma_2^P}$*-complete.*

4 Model Checking with Alternating Transition Systems

We have shown that model checking ATL over CGS is $\mathbf{\Sigma_2^P}$-complete in the previous section, when the size of models is defined in terms of the number of states, and the number of agents is a parameter of the problem. However, the transition function in a CGS refers to choices that are abstract, while in alternating transition systems the function already encodes some information about possible outcomes of actions. One obvious advantage is that, in an ATS, the transition function is already represented in a compact way: for n states, k agents and at most d decisions per agent and state, the size of function δ is $O(n^2kd)$, while the transition function in a CGS may require $O(nd^k)$ memory cells in general. In this section, we show that using ATS implies also some advantage in terms of model checking complexity: it still sits in the nondeterministic polynomial hierarchy, but it is "only" $\mathbf{NP}$-complete. First, we demonstrate that the model checking is in $\mathbf{NP}$ in Section 4.1. Then, in Sections 4.2 and 4.3, we define a variant of the Boolean satisfiability problem that we call "single false clause SAT" (*sfc-SAT*), prove that it is $\mathbf{NP}$-complete, and present a reduction of *sfc-SAT* to the model checking problem.

Modelling systems via ATS is usually troublesome in practice, mostly due to the "singleton" requirement. In Section 4.4, we point out that, if we relax the requirement and allow for nondeterministic ATS's, the model checking problem remains $\mathbf{NP}$-complete – that is, we obtain the same model checking complexity for a strictly larger class of models.

4.1 Model Checking ATL over Alternating Transition Systems is $\mathbf{NP}$-easy

Unlike in concurrent game structures, choices in alternating transition systems already contain some information about which states can possibly be achieved through them. More precisely, α includes *all* the states that can be achieved through α. Had it contained *only* such states, checking if it enforces φ would have been easy (it would have been sufficient to check whether φ holds in all $q' \in \alpha$). However, the latter condition is not true in general. [15] introduces the notion of a *tight* ATS: all states q' to which no transition exists from q are removed from agents' choices at q (i.e. from the elements of $\delta(q, a)$ for all $a \in \mathbb{A}\mathrm{gt}$). Still, this is not enough for our purposes, because $\alpha \in \delta(q, a)$ may include states that are reachable from q in general, but not by executing α. In this section, we propose a stronger notion of tightness, and show a nondeterministic algorithm to model check ATL formulae over such ATS's. We also present a nondeterministic algorithm to "tighten" an ATS, and point out how these algorithms can be combined to obtain a procedure that model checks ATL formulae over arbitrary ATS's in nondeterministic polynomial time. In the following, we assume without loss of generality that $A = \{a_1, ..., a_r\}$ for some $r \leq k$.

Definition 1 *Let $\alpha_A = \langle \alpha_1, ..., \alpha_r \rangle$ be a collective choice of A at q, i.e. $\alpha_i \in \delta(q, a_i)$.*

function $tighten(M)$**;**
For every $a_i \in \mathbb{A}\mathrm{gt}$, $q \in St$, $\alpha_i \in \delta(q, a_i)$, and $q' \in \alpha_i$: ■ guess the "opposition" responses $\alpha_1, ..., \alpha_{i-1}, \alpha_{i+1}, ..., \alpha_k$; ■ **if** $q' \notin \alpha_1 \cap ... \cap \alpha_k$ **then** remove q' from α_i;

Figure 6: Algorithm for "tightening" alternating transition systems

function $pre_3(M, A, Q1)$**;**
■ $pre := \emptyset$; ■ for every $q \in St$: – guess $\alpha_a \in \delta(q, a)$ for each $a \in A$; – **if** $\bigcap_{a \in A} \alpha_a \subseteq Q1$ **then** $pre := pre \cup \{q\}$; ■ return pre;

Figure 7: New pre-image function for model checking ATL over alternating transition systems

State q' is α_A-reachable from q if there is a combination of responses from the rest of agents: $\alpha_{r+1}, ..., \alpha_k$, $\alpha_i \in \delta(q, a_i)$ such that $q' \in \alpha_1 \cap ... \cap \alpha_k$.

Definition 2 ATS M *is* strongly tight *if, for each $q \in St, a \in \mathbb{A}\mathrm{gt}$, we have that for every $q' \in \alpha_a \in \delta(q, a)$, q' is α_a-reachable from q.*

Lemma 14 *Let M be strongly tight, $\alpha_1, ..., \alpha_r$ be choices of $a_1, ..., a_r$ at q, and $q' \in \alpha_1 \cap ... \cap \alpha_r$. Then q' is $\langle \alpha_1, ..., \alpha_r \rangle$-reachable from q.*

Every ATS can be made strongly tight via the procedure in Figure 6. Moreover, ATL formulae can be model-checked over strongly tight ATS's via the original ATL model checking algorithm from Figure 3, with function $pre(A, Q)$ implemented as in Figure 7. We observe that – if we assign numbers $1, ..., |\delta(q, a)|$ to choices from $\delta(q, a)$ for all q, a at the beginning, so that the choices are further identified by abstract labels rather than their content – all the "guessing" operations are independent from each other. Thus, we can apply the same trick as in Section 3.3, and guess *all* the necessary information beforehand. The size of the witness is $O(n^2k^2d + nkl)$, hence we obtain an $\mathbf{NP}$ algorithm for the model checking.

Proposition 15 *Model checking formulae of* ATL *over alternating transition systems is* $\mathbf{\Sigma_2^P}$*-easy.*

4.2 Single False Clause SAT

Definition 3 [Single false clause SAT (**sfc-SAT**)]. *We define the following variant of the SAT problem.*

Input: *(1) n clauses: $C_1, ..., C_n$, in k propositions: $p_1, ..., p_k$ such that for each valuation of $p_1, ..., p_k$,* exactly one clause is false*;*
(2) numbers $m \leq n$, $r \leq k$.

Problem: Is there a valuation of $p_1, ..., p_r$ such that all clauses $C_1|_r, ..., C_m|_r$ are satisfied? *Clause $C|_r$ is obtained from clause C by deleting all literals that refer to propositions $p_{r+1}, ..., p_k$ (i.e., we keep only the literals up to r).*

Remark 16 *Obviously,* sfc-SAT *is in* $\mathbf{NP}$ *(it is sufficient to guess a valuation and check whether it is a good one).*

In order to show that *sfc-SAT* is $\mathbf{NP}$-hard, we show that 3-SAT can be reduced to it. In 3-SAT, we are given m clauses $C_1, ..., C_m$ over r propositions $p_1, ..., p_r$ such that each clause C_i contains at most three literals: $C_i = l_{i,1} \lor l_{i,2} \lor l_{i,3}$ ($l_{i,j}$ are p_l or $\neg p_l$, $1 \leq i \leq m$). This special instance of the satisfiability problem is also $\mathbf{NP}$-complete [32]. Note that the m and the r are the respective numbers occurring as inputs in Definition 3. To show that 3-SAT can be reduced to *sfc-SAT*, we demonstrate that there are propositions $p_{r+1}, \ldots, p_k$, and clauses $C'_1, ..., C'_n$, with $m \leq n$, $C_i \subseteq C'_i$ and $C'_i|_r = C_i$ for $i \leq m$, such that for each valuation of $p_1, \ldots, p_k$, exactly one of C'_i is false.

What does the last condition mean for a set of clauses $C'_1, ..., C'_n$? Basically, it means that these clauses represent all 2^k possibilities of choosing truth values for $p_1, \ldots, p_k$. So, the problem in the reduction is to *extend the given clauses by new variables* and to *add new clauses*. This has to be done so that the length of the new problem is still polynomial in the length of the given 3-SAT instance.

We assume without loss of generality that none of $C'_1, ..., C'_m$ contains a complementary pair of literals (otherwise the clause would be satisfiable under all valuations and could be safely discarded as it does not matter for the overall satisfiability problem). In order to extend clauses $C_1, ..., C_m$ in an appropriate way, we use auxiliary formulae α_i and β, defined in the following way:

α_i**:** We construct formulae α_i stating that *a selected clause number is* $i \leq m$. To be more precise, we introduce $t := \lceil \log m \rceil$ new variables $y_1, \ldots, y_t$ and define conjunctions α_i ($i = 1, \ldots, m$) over these variables as follows (this idea is due to Thomas Eiter [11]). We write each number $1, \ldots, m$ in binary and represent each (of the t) digits by the new variables (a 1 is represented by the variable itself, a 0 by the negation of the variable). The i'th digit is then represented by y_i if it is 1 and by $\neg y_i$ if it is 0. Thus, for each valuation of the new variables, only one conjunction α_i can be true, namely the one representing the number coded in the binary representation.

Note that we can also represent numbers greater than m (up to the next power of 2, namely 2^t). These conjunctions do not correspond to the m original clauses from the 3-SAT problem. In our reduction, we have to distinguish between them. Therefore we introduce a formula β in the next step.

β**:** We construct a formula β stating that the selected clause number is less than or equal to m. Thus, β satisfies the following equivalences: $\beta \Leftrightarrow \bigvee_{i=1}^{m} \alpha_i \Leftrightarrow \bigwedge_{i=m+1}^{2^{\lceil \log m \rceil}} \neg\alpha_i$.

We therefore define a set of clauses β, which describe all valuations corresponding to numbers strictly greater than m. Thus we have:

$$\beta \Leftrightarrow \bigwedge_{i=1}^{m} \neg\alpha_i.$$

Realising β as a set of clauses is simple: we just take α_m and check that they coincide on an initial segment and then a negated variable occurs (where in α_m a positive variable is located).

β can also be written as a set of clauses

$$\beta \Leftrightarrow \bigwedge_{i=m+1}^{2^{\lceil \log m \rceil}} \neg\alpha_i.$$

Note that the last formula is a set clauses (because all $\neg\alpha_i$ are clauses), and hence we need at most $2^{\lceil \log m \rceil} - m$ many clauses to represent β (which is never more than m). We denote these clauses by $C_1^\beta, \ldots, C_m^\beta$. Each clause C_j^β states, that *the selected clause has not the number* $m + j$.

In the following we sometimes use β to represent the (at most) m clauses $C_1^\beta, \ldots, C_m^\beta$.

Extending the clauses: For each $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$ we construct the *remaining* 7 clauses (all parities of the 3 variables) and add $\neg\alpha_i$. So, for each C_i we get 8 clauses $C'_{i,0}, \ldots C'_{i,7}$, where $C'_{i,0} = C_i \vee \neg\alpha_i$ and $(C'_{i,0} \wedge \ldots \wedge C'_{i,7}) \Leftrightarrow \neg\alpha_i$. Note, again, that $\neg\alpha_i$ is always a clause. We observe also that the m clauses $C_1, ..., C_m$, which we originally started with (as an instance of 3-SAT), are, by construction, exactly $C'_{1,0}|_r, C'_{2,0}|_r, \ldots, C'_{m,0}|_r$.

Reduction: The (at most) $s := m + m \times 8$ clauses:

$$C_1^\beta, \ldots, C_m^\beta, \qquad \text{and} \qquad C'_{i,j} \;\; (1 \leq i \leq m, 0 \leq j \leq 7),$$

over $k = r + \lceil \log m \rceil$ variables, represent an instance of *sfc-SAT*, such that if we choose $m \leq n$ and $r \leq k$, then we get the 3-SAT problem we started with.

Why are the clauses above an instance of *sfc-SAT*? The fact that we get back the 3-SAT problem has already been shown. It is also obvious that the constructed instance is polynomial in the size of the instance we started with. So it remains to show that for each valuation of all the variables, *exactly one clause is false*. Let a valuation be given. We must consider two cases:

1. Exactly one of the $\alpha_1, \ldots, \alpha_m$ is true, say α_{i_0} (this is decided by the newly introduced variables). Then all clauses $C'_{i,j}$ with $i \neq i_0$ are true (because $\neg\alpha_i$ is true and it occurs as a disjunct in all these clauses). Of the 8 clauses $C'_{i_0,j}$ $(0 \leq j \leq 7)$, exactly one is false, namely the one contradicting the valuation of the three old variables occurring in the original C_i (note that all possibilities are covered with the 8 cases). Clearly, β (i.e all clauses C'^{β}_j) is true as well.

2. None of the $\alpha_1, \ldots, \alpha_m$ is true. But then all clauses $C'_{i,j}$ are true and only β is false, i.e. exactly one of the clauses C'^{β}_j.

These are all the cases, because α_i (resp. C'^{β}_j) are pairwise inconsistent by construction: any two different conjunctions α_i, α_j (resp. $C'^{\beta}_i, C'^{\beta}_j$) with $i \neq j$ contain at least one pair of complementary literals. This gives us the following result:

Proposition 17 *Sfc-SAT is* $\mathbf{NP}$*-complete.*

4.3 Reduction of *sfc-SAT* to ATL Model Checking over ATS

To obtain the reduction, we construct an ATS M with states $St = \{q_0, C_1, ..., C_n\}$, i.e. one state per clause plus an initial state. Next, we "simulate" propositions $p_1, ..., p_k$ with agents $a_1, ..., a_k$. Each agent "declares" his proposition true or false in the initial state q_0. Thus, agent a_i has two available choices at q_0: to declare p_i true or to declare p_i false; a choice of a_i is represented with the set of clauses that are *not* made true by setting the value of p_i in this particular way. For example, for clauses $C_1 = p_1 \vee \neg p_2, C_2 = p_2$, a_1's choices are represented as $\{C_2\}, \{C_1, C_2\}$: if p_1 is set to true, only C_2 can be false, but if p_1 is set to false, both C_1, C_2 can remain unsatisfied. Choices and transitions at states $C_1, ..., C_m$ do not really matter. There is only one atomic proposition, $\mathsf{therest}$, with $\pi(\mathsf{therest}) = \{C_{m+1}, ..., C_n\}$.

Note that each combination of choices from $a_1, ..., a_k$ at q_0 corresponds to a single valuation of $p_1, ..., p_k$, and vice versa. Moreover, a clause is not satisfied by a valuation iff no proposition "makes" it true. Thus, the set of clauses, unsatisfied by a valuation, is equal to the intersection of sets of clauses that are not "made" true by each single proposition. By definition of *sfc-SAT*, such an intersection is always a singleton, which proves that M is indeed an ATS.

Lemma 18 *There is a valuation of* $p_1, ..., p_r$ *such that all clauses* $C_1|r, ..., C_m|r$ *are satisfied iff* $M, q_0 \models \langle\langle a_1, ..., a_r \rangle\rangle \bigcirc \mathsf{therest}$.

Proof. **[⇒]** Suppose that there is such a valuation of $p_1, ..., p_r$. Thus, regardless of the actual valuation of $p_{r+1}, ..., p_k$, clauses $C_1, ..., C_t$ must be true, and hence for each valuation of $p_{r+1}, ..., p_k$, the single unsatisfied clause must be among $C_{t+1}, ..., C_n$. Rewriting it in terms of the ATS M: there is a collective choice of $a_1, ..., a_r$ such that, for every tuple of choices from the other agents, the resulting next state must be among $C_{t+1}, ..., C_n$. In consequence, $M, q_0 \models \langle\langle a_1, ..., a_r \rangle\rangle \bigcirc \mathsf{P}$.

[⇐] Let $M, q_0 \models \langle\langle a_1, ..., a_r \rangle\rangle \bigcirc \mathsf{P}$. Thus, there is a collective choice $S_{\{a_1,...,a_r\}}$ such that, for every tuple of choices from the other agents, the resulting next state is always among $C_{t+1}, ..., C_n$. We take the valuation of $p_1, ..., p_r$ that corresponds to this $S_{\{a_1,...,a_r\}}$. By the shape of the construction, each $C_i \in \{C_1, ..., C_t\}$ must be true for each valuation of $p_{r+1}, ..., p_k$. In particular, C_i is true for the valuation $p_{r+1} = \bot, ..., p_k = \bot$. Thus, $C_i|r$ is also true. ■

Note that the reduction can be done in time polynomial in n, k. Computing the agents' choice sets is the hardest point here, and it can be done in time $O(k^2 n)$. The resulting model includes $n + 1$ states, k agents, and $d = 2$ choices per agent per state – and the length of the resulting formula is $l = r + 2 \le k + 2$, which concludes the reduction, and proves that the model checking problem is $\mathbf{NP}$-hard. Thus, we have the following.

Theorem 19 *Model checking* ATL *formulae over* ATS *is* $\mathbf{NP}$*-complete.*

4.4 Model Checking with Nondeterministic Transition Systems

Alternating transition systems were proposed as models for open computational systems, and the way in which the transition function is constructed reflects this intention. The problem with ATS's is that they are *not* modular, partly due to the "singleton intersection" requirement: legality of a choice cannot be defined in isolation from the rest of the choices model. Adding another process to the system usually requires thorough reconstruction of the model: in particular, new states must be added, and agents' choices extended so that *every* intersection is again a singleton. We suggest that the requirement can be relaxed, yielding a more general (and more flexible) class of models with the same ATL model checking complexity. To show this, we define *non-deterministic alternating transition systems* (NATS) in the same way as ATS, except that no requirement on function δ is imposed.[2] Obviously, model checking ATL formulae over NATS is $\mathbf{NP}$-hard, because ATS are special cases of NATS. Moreover, the model checking algorithm, depicted in Section 4.1, can be applied to NATS as well.

Theorem 20 *Model checking* ATL *formulae over* NATS *is* $\mathbf{NP}$*-complete.*

This suggests that using the more general class of NATS may be beneficial for most purposes: we can get rid of the rigid and highly inconvenient "singleton" requirement

[2] Traditionally, the transition relation is required to be serial in models of temporal logic, in order to make sure that the time "flows forever". This can be ensured by requiring that, for every tuple of choices $\alpha_i \in \delta(q, a_i)$, the intersection $\alpha_1 \cap ... \cap \alpha_k$ is non-empty. Our argument in this section is valid for such a variant of NATS, too.

case $\varphi = \langle\langle A\rangle\rangle\psi_1\,\mathcal{W}\,\psi_2$ **:**
$Q_1 := mcheck(M,\psi_1) \cup mcheck(M,\psi_2)$;
$Q_2 := St$;
$Q_3 := mcheck(M,\psi_2)$;
while $Q_2 \neq Q_1$
do $Q_2 := Q_1$; $\quad Q_1 := Q_2 \setminus \big((St \setminus pre(M,A,Q_2)) \setminus Q_3\big)$ **od**;
return Q_1

Figure 8: Subroutine for model checking "weak until"

without any computational cost! But, as we already pointed out in Sections 2.1 and 2.2, the existential path quantifier E from CTL cannot be fully embedded in ATL, when the latter has its semantics defined over NATS. This looks as a serious shortcoming in terms of expressivity at the first glance. However, we observe that we can deal with this problem by adding another temporal operator to the language of ATL. The operator we propose to add is the "weak until" operator $\mathcal{W}$, known in temporal logic for a long time [28], although not as popular as the "strong until" $\mathcal{U}$. Formula $\varphi\,\mathcal{W}\,\psi$ is meant to express the fact that if ψ becomes eventually true, then φ holds until the first occurrence of ψ, otherwise φ holds for ever. The formal semantics of "weak until" can be defined as follows:

$M, q \models \langle\langle A\rangle\rangle\varphi\,\mathcal{W}\,\psi$ iff there exists S_A such that, for every $\lambda \in out(S_A, q)$: (1) there is $i \geq 0$ for which $M, \lambda[i] \models \psi$, and $M, \lambda[j] \models \varphi$ for every $0 \leq j < i$, or (2) $M, \lambda[j] \models \varphi$ for every $j \geq 0$.

In Figure 8, we present a simple extension of the model checking from Figure 3 that deals with "weak until" formulae in a way analogous to model checking $\langle\langle A\rangle\rangle\Box\varphi$ and $\langle\langle A\rangle\rangle\varphi\,\mathcal{U}\,\psi$. The model checking algorithm from Section 4.1 can be augmented in the same way. It is easy to see that the complexity of the algorithms stays the same as before. Moreover, we show that adding $\langle\langle A\rangle\rangle\varphi\,\mathcal{W}\,\psi$ to ATL allows to express the full power of CTL (and more), through the following translation:

$$
\begin{aligned}
\mathsf{A}\bigcirc\varphi &\equiv \langle\langle\varnothing\rangle\rangle\bigcirc\varphi \\
\mathsf{A}\varphi\,\mathcal{U}\,\psi &\equiv \langle\langle\varnothing\rangle\rangle\varphi\,\mathcal{U}\,\psi \\
\mathsf{A}\varphi\,\mathcal{W}\,\psi &\equiv \langle\langle\varnothing\rangle\rangle\varphi\,\mathcal{W}\,\psi \\
\mathsf{E}\bigcirc\varphi &\equiv \neg\mathsf{A}\bigcirc\neg\varphi \\
\mathsf{E}\varphi\,\mathcal{U}\,\psi &\equiv \neg\mathsf{A}(\neg\psi)\,\mathcal{W}\,(\neg\varphi\wedge\neg\psi) \\
\mathsf{E}\varphi\,\mathcal{W}\,\psi &\equiv \neg\mathsf{A}(\neg\psi)\,\mathcal{U}\,(\neg\varphi\wedge\neg\psi) \\
\Diamond\varphi &\equiv \top\,\mathcal{U}\,\varphi
\end{aligned}
$$

$$\Box\varphi \quad\equiv\quad \varphi\mathcal{W}\bot$$

Note that formulae $\langle\langle A\rangle\rangle\Box\varphi$ do not have to be included in the definition of ATL explicitly any more, since the $\Box$ operator can be derived from $\mathcal{W}$.

Theorem 21 ATL *with "weak until" covers the full expressive power of* CTL *even when nondeterministic alternating transition systems are used as models.*

Moreover, model checking ATL *with "weak until" over* NATS *(nondeterministic* ATS*) is:*

1. **PTIME**-*complete (linear time) with respect to the number of transitions in the model and the length of the formula,*

2. **NP**-*complete with respect to the number of states, agents and decisions in the model and the length of the formula.*

Finally, we observe that ATS have already been used in the work on implementing symbolic model checking for ATL [24], probably because of the compact representation of the transition function.[3] We proved in this section that using ATS offers also some computational advantage over CGS. Theorem 21 suggests that *designing* ATS does not have to be such a painstaking process.

5 Turning Game Models Turn-Based

In this section, we demonstrate how strategic ability in arbitrary ATS's can be translated into strategic ability in turn-based systems. More precisely, we show how, for an arbitrary alternating transition system M, a turn-based system M' can be constructed, so that a combination of *choices* in M corresponds to a combination of *strategies* in a fragment of M'. We then propose a translation of ATL formulae into ATL$^+$ formulae, such that the original formula holds in M, q if, and only if, the translated formula holds in M', q. Finally, we point out that the latter can be model-checked in nondeterministic polynomial time, and provide another (slightly more general) proof that the problem is in NP.

The translation of models is independent from the translation of formulae in our construction, which allows for "pre-compiling" models when one wants to check various properties of a particular multi-agent system.

[3] The authors define the semantics of ATL in terms of concurrent game structures, but the model checking algorithm they present uses *postconditions* to specify the possible outcomes of a choice. A postcondition is taken to be simply a set of states, and choices by different agents executed in parallel lead to the state from the intersection of the postconditions (it is even assumed that the intersection must be a singleton).

5.1 Translation of Models

Let $M = \langle \mathbb{A}\mathrm{gt}, St, \Pi, \pi, \delta \rangle$ be an ATS. We construct a turn-based ATS $M' = \langle \mathbb{A}\mathrm{gt}', St', \Pi', \pi', \delta' \rangle$ as follows:

- $\mathbb{A}\mathrm{gt}' = \mathbb{A}\mathrm{gt} \cup \{v\}$: we add an additional agent v ("verifier") to the original set of players. Verifier helps to find out the right outcome state, given the choices from all agents (i.e. the sole state which belongs to the intersection of their choices);
- $St' = St \cup \bigcup_{a \in \mathbb{A}\mathrm{gt}} (dec(a) \cup exec(a) \cup outcome(a))$, where:
  - $dec(a) = \{q^a \mid q \in St\}$ are the "dummy states" from which agent a's decisions are simulated; by x^ρ, we will denote a copy of item x, labelled with superscript ρ.
  - $exec(a) = \{q^{a,S} \mid q \in St, S \in \delta(q,a)\}$ simulate the situations between a's decision making and the execution of a decision.
  - $outcome(a) = \{q^{a,q'} \mid q \in St, q' \in \bigcup_{S \in \delta(q,a)} S\}$ are the dummy states that simulate possible outcomes of a's decisions.
- $\Pi' = \{\overline{\mathsf{q}} \mid q \in St\} \cup \{\overline{\mathsf{real}}, \overline{\mathsf{choice}}, \overline{\mathsf{out}}\}$. Proposition $\overline{\mathsf{real}}$ marks the original, "real" states from M; $\overline{\mathsf{choice}}$ labels the dummy states that simulate situations before and after a choice, $\overline{\mathsf{out}}$ marks the *final* outcome states before the next "real" state is reached, and $\overline{\mathsf{q_i}}$ mark "outcome" dummy states that refer to a transition ending up in state q_i. Thus:
  - $\pi'(\overline{\mathsf{real}}) = St$,
  - $\pi'(\overline{\mathsf{choice}}) = \bigcup_{a \in \mathbb{A}\mathrm{gt}} (dec(a) \cup exec(a))$,
  - $\pi'(\overline{\mathsf{out}}) = outcome(a_k)$,
  - $\pi'(\overline{\mathsf{q_i}}) = \{q^{a,q_i} \mid q^{a,q_i} \in outcome(a), a \in \mathbb{A}\mathrm{gt}\}$.
- The "decision" states are "owned" by the decision making players; the rest of the states is owned by Verifier:
  - $\delta(q,v) = \{St'\}$ for $q \in dec(a), a \in \mathbb{A}\mathrm{gt}$ (Verifier has no control in the "decision" states);
  - $\delta(q,a) = \{St'\}$ for $q \notin dec(a)$ (a has no control in the states outside $dec(a)$).
- Choices of the original agents remain the same as in M, but they are split between "choice" states. Verifier makes substantial choice only at the "execution" dummy states. Transitions from the "outcome" dummy states are automatic, and lead to the decision node of the next player. Choices executed by agents at decision nodes lead to their corresponding execution states, and Verifier's actions at execution nodes lead to their corresponding outcome nodes.

– $\delta(q^a, a) = \{\{q^{a,Q_1}\}, ..., \{q^{a,Q_i}\}\}$ for $q^a \in dec(a)$, and $\delta(q, a) = \{Q_1, ..., Q_i\}$;
– $\delta(q^{a,S}, v) = \{\{q^{a,q_1}\}, ..., \{q^{a,q_i}\}\}$ for $q^{a,S} \in exec(a)$ and $S = \{q_1, ..., q_i\}$.
– $\delta(q^{a_i,q_j}, v) = \{\{q^{a_{i+1}}\}\}$ for $i < k$, and $\delta(q^{a_k,q_j}, v) = \{\{q^{a_k}\}\}$.

Example 3 *Consider a fragment of the alternating transition system M_2, depicted in Figure 2. The fragment of the resulting* ATS *M_2', that refers to the transitions starting from q_0, is shown in Figure 9. (Remember, we use symbols α_1, α_2 and β_1, β_2 as shorthand for the choices to make the example easier to read, but in fact these are sets of states and not abstract labels.) The collective strategy of $\{a, b\}$, that corresponds to the combination of choices $\langle\alpha_1, \beta_2\rangle$ in the original* ATS*, is marked with bold arrows. The only Verifier's response, that yields a path with exactly one $\overline{\mathsf{q}}_\mathsf{i}$ proposition holding along it, is also indicated.*

Note that, for each state q in M, the transformation of the outgoing transitions requires that we process all the choices from $\delta(q, a)$ once; we must also process the "contents" of every choice (i.e. all the states included in the choice) – but only once, too. Moreover, the resulting substructure includes at most $O(kdn)$ outgoing transitions per node.

Proposition 22 *The translation of M can be done in time $O(n^2kd)$, and M' includes $m' = O(n^2kd)$ states.*

5.2 Translation of Formulae

Let φ, ψ be ATL formulae, whose interpretations in M are $[\![\varphi]\!], [\![\psi]\!]$ respectively.[4] Then we define the translation of complex formulae in the following manner:

$$
\begin{aligned}
tr_M(\neg\varphi) &= \neg[\![\varphi]\!] \\
tr_M(\varphi \wedge \psi) &= [\![\varphi]\!] \wedge [\![\psi]\!] \\
next_M(\varphi) &= \neg \bigvee_{q_i \notin [\![\varphi]\!]} (\overline{\mathsf{q}}_\mathsf{i} \vee \overline{\mathsf{real}} \vee \overline{\mathsf{choice}})\,\mathcal{U}\ \overline{\mathsf{out}} \\
tr_M(\langle\!\langle A\rangle\!\rangle\bigcirc\varphi) &= \langle\!\langle A\rangle\!\rangle next_M(\varphi) \\
tr_M(\langle\!\langle A\rangle\!\rangle\Box\varphi) &= [\![\varphi]\!] \wedge \langle\!\langle A\rangle\!\rangle\Box(\overline{\mathsf{real}} \rightarrow \langle\!\langle\emptyset\rangle\!\rangle next_M(\varphi)) \\
tr_M(\langle\!\langle A\rangle\!\rangle\varphi\,\mathcal{U}\,\psi) &= [\![\psi]\!] \vee ([\![\varphi]\!] \wedge \langle\!\langle A\rangle\!\rangle(\overline{\mathsf{real}} \rightarrow \langle\!\langle\emptyset\rangle\!\rangle next_M(\varphi))\,\mathcal{U}\ [\![\psi]\!]).
\end{aligned}
$$

The idea is as follows: the paths that matter are the ones where only a single proposition $\overline{\mathsf{q}}_\mathsf{i}$ occurs in each subpath between two subsequent "real" states – they correspond to intersections of the agents' choices that can be found along the subpath. For $\langle\!\langle A\rangle\!\rangle\bigcirc\varphi$,

[4] We will abuse the notation slightly by using $[\![\varphi]\!]$ to denote also $\bigvee_{q_i \in [\![\varphi]\!]} \overline{\mathsf{q}}_\mathsf{i}$, a formula that holds exactly in the states from $[\![\varphi]\!]$.

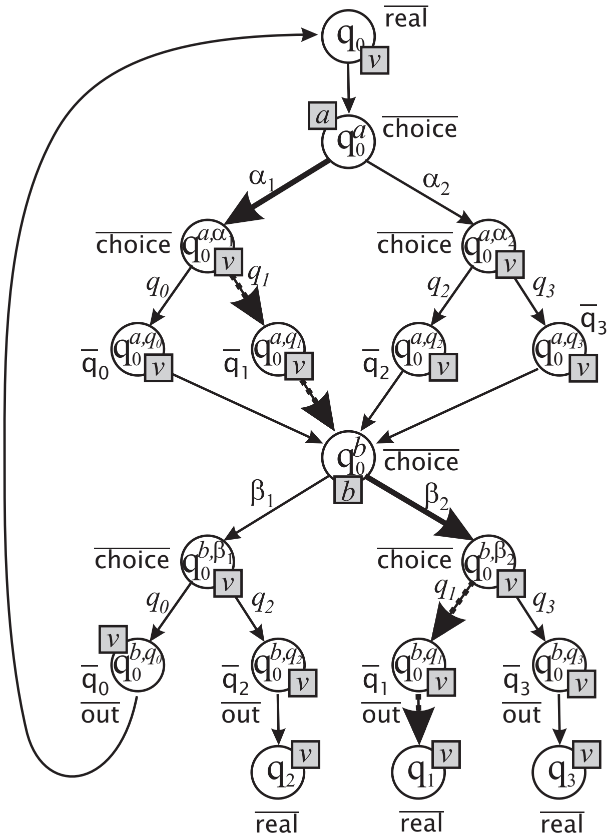


Figure 9: A fragment of M_1': simulation of outgoing transitions from q_0

we want to make sure that A have a strategy to enforce that all such subpaths until the next "real" node refer to states from $[\![\varphi]\!]$. In other words, A must have a strategy such that *no* initial subpath occurs that refers to some $q_i \notin [\![\varphi]\!]$. For $\langle\langle A\rangle\rangle\Box\varphi$, the same must hold for subpaths after *each* "real" node etc. Note that $tr_M(\Phi)$ is a formula of ATL$^+$, since it includes Boolean combinations of temporal formulae.[5] The following proposition states that the translation is correct.

Proposition 23 *Let φ be an* ATL *formula that does not include special propositions* $\overline{\mathsf{real}}$, $\overline{\mathsf{out}}$, $\overline{\mathsf{choice}}$ *and* $\overline{\mathsf{q}}_\mathsf{i}$. *Let M be an* ATS, *and q a state in M. Then:*

$$M, q \models \Phi \quad \textit{iff} \quad M', q \models tr_M(\Phi).$$

[5] We assume that this is the "memoryless" version of ATL$^+$, with strategies represented as functions from *states* to choices.

Proof. We prove the proposition for the case $\Phi \equiv \langle\langle A\rangle\rangle \bigcirc \varphi$. The other cases follow from respective fixpoint characterisations of $\langle\langle A\rangle\rangle \Box\varphi$ and $\langle\langle A\rangle\rangle \varphi \mathcal{U} \psi$.

[⇒] Let $M, q \models \langle\langle A\rangle\rangle \bigcirc \varphi$, $A = \{a_1, ..., a_r\}$. Suppose that $M', q \not\models \langle\langle A\rangle\rangle \neg \bigvee_{q_i \notin [\![\varphi]\!]} (\overline{\mathsf{q_i}} \vee \overline{\mathsf{real}} \vee \overline{\mathsf{choice}}) \mathcal{U}\ \overline{\mathsf{out}}$. Note that $\overline{\mathsf{out}}$ holds for the first time exactly after $3k$ transitions from q in M'. Thus, for every strategy S'_A in M' there is a path $\Lambda \in out(q, S'_A)$, and a state $q_i \notin [\![\varphi]\!]$, such that $M, \Lambda[j] \models (\overline{\mathsf{q_i}} \vee \overline{\mathsf{real}} \vee \overline{\mathsf{choice}})$ for all $j = 0, ..., 3k-1$. We take any strategy S_A in M, find the corresponding S'_A with $s'_a(q^a) = s_a(q)$ for $a \in A$, and then we take the above Λ and q_i. We set the choices of the opponents in M, q to $s_{a_j}(q) = \sigma$ such that $\Lambda[3j-2] = q^{a_j,\sigma}$, $a_j \notin A$. By construction, $q_i \in s_{a_1}(q) \cap ... \cap s_{a_k}(q)$, which gives a contradiction.

[⇐] Similarly: we take the "winning" strategy in M', construct the corresponding strategy in M (or rather its relevant part for state q), and show that no combination of responses from $a_{r+1}, ..., a_k$ can lead to a state $q' \notin [\![\varphi]\!]$. ■

Example 4 *Consider models M_2 and M'_2 again. Formula $\langle\langle a\rangle\rangle \bigcirc (\mathsf{p_1} \vee \mathsf{p_2})$ holds in M_2, q_0, and indeed $M'_2, q_0^a \models \langle\langle a\rangle\rangle \neg\big((\overline{\mathsf{q_2}} \vee \overline{\mathsf{real}} \vee \overline{\mathsf{choice}}) \mathcal{U}\ \overline{\mathsf{out}}\ \vee (\overline{\mathsf{q_3}} \vee \overline{\mathsf{real}} \vee \overline{\mathsf{choice}}) \mathcal{U}\ \overline{\mathsf{out}}\big)$. On the other hand, $M'_2, q_0 \not\models \langle\langle a\rangle\rangle \bigcirc \mathsf{p_1}$, and also $M'_2, q_0^a \not\models \langle\langle a\rangle\rangle \neg\big((\overline{\mathsf{q_1}} \vee \overline{\mathsf{real}} \vee \overline{\mathsf{choice}}) \mathcal{U}\ \overline{\mathsf{out}}\ \vee (\overline{\mathsf{q_2}} \vee \overline{\mathsf{real}} \vee \overline{\mathsf{choice}}) \mathcal{U}\ \overline{\mathsf{out}}\ \vee (\overline{\mathsf{q_3}} \vee \overline{\mathsf{real}} \vee \overline{\mathsf{choice}}) \mathcal{U}\ \overline{\mathsf{out}}\big)$.*

Proposition 24 *The length of $tr_M(\varphi)$ is $l' = O(n + l)$, where l is the length of φ, and n is the number of states in M.*

The following nondeterministic algorithm can be used to model check formula $\langle\langle A\rangle\rangle\varphi$ of the "memoryless" ATL* in model M':

1. Guess the collective strategy S_A. Note that the size of S_A is $O(nkd)$;
2. "Trim" model M', removing all A's choices that do not appear in S_A. As M' is turn-based, the operation requires only $O(nkd)$ steps, and yields a turn-based ATS M'' with no more states and transitions than M';
3. Model-check CTL* formula $\mathsf{A}\varphi$ in M''.

Note that $\mathsf{A}\ next_M(\varphi) \Leftrightarrow \neg\mathsf{E} \bigvee_{q_i \notin [\![\varphi]\!]} (\overline{\mathsf{q_i}} \vee \overline{\mathsf{real}} \vee \overline{\mathsf{choice}}) \mathcal{U}\ \overline{\mathsf{out}} \Leftrightarrow \neg \bigvee_{q_i \notin [\![\varphi]\!]} \mathsf{E}(\overline{\mathsf{q_i}} \vee \overline{\mathsf{real}} \vee \overline{\mathsf{choice}}) \mathcal{U}\ \overline{\mathsf{out}}$, which is a formula of "vanilla" CTL, and can be model-checked in deterministic polynomial time.[6] Note also that an array of strategies for all the cooperation modalities occurring in a complex formula can be guessed *before* the translation of the formula (as strategy $s_a(q) = \alpha$ in M transformed to an equivalent strategy $s'_a(q^a) = \{\{q^{a,\alpha}\}\}$ in M'). The size of the witness is still $O(nkdl)$, which gives us the following.

Corollary 25 *Model checking of an ATL formula φ in an ATS M is NP-easy in n, k, d, l.*

[6] We thank an anonymous reviewer for pointing this out.

Thus, we obtained a proof of NP-easiness, alternative to [19]. We want to emphasise that the above algorithm is somewhat more general than the one in [19], because it does *not* employ "tightening" of the model. In principle, an equivalent tight model exists for every ATS if we consider alternating transition systems in isolation. However, the same does not have to hold when we extend ATL and ATS with additional modalities. For instance, for an ATL extension that handles incomplete information, we may want to require that a single strategy specifies identical choices in indistinguishable states (cf. [37]), which means that a choice must include all the states that are considered possible outcomes by an agent in a given situation, and not only the ones that can *physically* occur [35]. In consequence, such a kind of alternating *epistemic* transition systems cannot be tight in most cases. The above algorithm is valid for all ATS, even for those which cannot be tightened in a given context.

6 Model Checking Strategic Abilities of Agents under Incomplete Information

In this section, we consider model checking of ATL *with imperfect* (or *incomplete*) *information*. Since no satisfying semantics based on alternating transition systems has been proposed so far for strategic abilities under incomplete information, we present our results for an extension of concurrent game structures only.

Schobbens [34] proved that ATL_{ir} model checking is intractable: more precisely, it is $\mathbf{NP}$-hard and $\mathbf{\Delta_2^P}$-easy (i.e., can be solved through a polynomial number of calls to an oracle for some problem in $\mathbf{NP}$) when the size of the model is defined in terms of the number of transitions. He also conjectured that the problem is probably $\mathbf{\Delta_2^P}$-complete.

This section contains several new results. Firstly, we show that ATL_{ir} model checking is in fact $\mathbf{NP}$-complete in the number of transitions in the model and the length of the formula. Secondly, we prove that the problem is $\mathbf{\Sigma_2^P}$-complete in the number of states, agents and decisions (per agent and state) in the model, and the length of the formula. Therefore, the problem sits in the same complexity class as model checking strategic abilities for *perfect* information games with respect to these parameters. We believe this is good news, as far as complexity is concerned, for agent logics dealing with incomplete information.

Finally, we point out that the difference between the perfect and imperfect information case lies in the modularity of strategies with respect to the property that the agents may want to enforce. For perfect information games, potential successfulness of substrategies is more independent and they can be computed (or guessed) incrementally, while imperfect information strategies refuse incremental analysis.

6.1 Existing Results

Model checking ATL_{ir} has been proved to be $\mathbf{NP}$-hard and $\mathbf{\Delta_2^P}$-easy in the number of transitions and the length of the formula [34]. The $\mathbf{NP}$-hardness follows from a reduction of the well known SAT problem: one can construct an imperfect information concurrent epistemic game structure M with states representing clauses and literals inside those clauses. At every "clause" state, the "clause" agent c chooses a transition to a state that represents literal l_i (i.e., either formula p_i or $\neg p_i$) that appears in this clause. "Literal" states for l_i are governed by agent a_i who can declare the underlying proposition p_i true or false. If it makes l_i false then we end up in a "sink" state q_{lose}; if it makes l_i true then the system proceeds to the next "clause" state (or, after the last clause, to state q_{win}). All the states referring to proposition p_i (or its negation) are indistinguishable for agent a_i, and therefore a_i has to make the same decision in all of them. Now, checking satisfiability of the set of clauses is equivalent to model checking of formula $\langle\langle a_1, ..., a_k\rangle\rangle\Diamond \mathsf{win}$ in the initial state of M, where win is a proposition that holds only in state q_{win}. Note that M is turn-based, i.e. at every state there is a single agent that decides upon the next transition. Moreover, it is easy to see that all the "literal" can be in fact governed by the same "literal" agent a: then the SAT problem reduces to model checking of formula $\langle\langle a\rangle\rangle_{ir}\Diamond \mathsf{win}$. Thus, the model checking problem for ATL$_{ir}$ is $\mathbf{NP}$-hard even for turn-based models with at most two agents.

The $\mathbf{\Delta_2^P}$-easiness can be demonstrated through the following observation. If the formula to be model checked is of the form $\langle\langle A\rangle\rangle_{ir}\varphi$ (φ being $\bigcirc\psi$, $\Box\psi$ or $\psi_1\,\mathcal{U}\,\psi_2$), where φ contains no more cooperation modalities, then it is sufficient to guess a strategy for A, "trim" the model by removing all transitions that will never be executed (according to this strategy), and model check CTL formula $\mathsf{A}\varphi$ in the resulting model. Thus, model checking an arbitrary ATL$_{ir}$ formula can be done by checking the subformulae iteratively, which requires a polynomial number of calls to an $\mathbf{NP}$ algorithm.

6.2 NP-completeness: Processing All Transitions

In [34], it was shown that an ATL$_{ir}$ formula can be model-checked via a polynomial number of calls to an $\mathbf{NP}$ oracle: as the size of a (collective) strategy is $\mathbf{O}(m)$, it is sufficient to process the formula recursively, "guessing" the right strategy every time a cooperation modality is encountered. We use a simple trick to show that it is enough to call the oracle only once: all the necessary strategies can be guessed *beforehand*. Note that the size of the witness is still polynomial in this case: more precisely, it is $\mathbf{O}(ml)$.

Theorem 26 *Model checking* ATL$_{ir}$ *is* $\mathbf{NP}$*-complete in the number of transitions in the model and the length of the formula.*

Proof. A nondeterministic algorithm that checks formula φ in model M is presented in Figures 10 and 11. Calls to $mcheck_{CTL}$ refer to any established CTL model-checker (e.g. [8]). As for the time necessary to carry out the procedure: guessing the strategies can be done in time $\mathbf{O}(ml)$, while "trimming" the model, checking CTL formulae, and getting rid of the states in which agents may not know that the strategy is successful, can all be done in time $\mathbf{O}(m)$ (recursively for subformulae). Thus, the algorithm terminates

function $mcheck_4(M, \varphi)$;
Returns the set of states in M, in which formula φ holds.

- assign cooperation modalities in φ with subsequent numbers $1, ..., c$;
  `// note that` $c \leq l$
  `// we denote the coalition from the` i`th coop. modality in` φ `as` $\varphi[i]$
- for every $i = 1, ..., c$, assign the agents in $\varphi[i]$ with numbers $1, ..., k_c$;
  `// note that` $k_c \leq k$ `and` $k_c \leq l$
  `// we will denote the` j`th agent in` A `with` $A[j]$
- guess an array $choice$ such that, for every $i = 1, ..., c$, $q \in St$, and $j = 1, ..., k_c$, we have that $choice[i][q][j] \in d_{\varphi[i][j]}(q)$, and for every $q' \in St$ such that $q \sim_{\varphi[i][j]} q'$ we have $choice[i][q][j] = choice[i][q'][j]$;
  `// now, the optimal choices for all coalitions in` φ `are guessed`
  `// note that the size of` $choice$ `is` $\mathbf{O}(ml)$
  `// by` $choice|_i$`, we denote array` $choice$ `with rows` $1, ..., i-1$ `removed`
- return $eval_4(M, \varphi, choice)$;

Figure 10: Nondeterministic algorithm for model checking formulae of ATL$_{ir}$; part I.

in time $\mathbf{O}(ml)$. Combining it with the $\mathbf{NP}$-hardness result [34], we obtain the theorem. ■

Note that the exhaustive deterministic algorithm that checks all possible strategies runs in time $\mathbf{O}(nd^{kn}l) = \mathbf{O}(n(m/n)^n l)$.

6.3 The Complexity Refined

One of the problems with model checking formulae of ATL is that the number of transitions m in a model is not bounded by n^2, and can be very large: more precisely, $m = \mathbf{O}(nd^k)$ where n is the number of states, k the number of agents, and d the maximal number of decisions per agent per state. Thus, m is exponential in k unless the model is turn-based or the number of agents is fixed. In Section 3, ATL model checking over concurrent game structures was proved to be $\mathbf{\Sigma_2^P} = \mathbf{NP^{NP}}$-complete when n, k, d, l are considered parameters of the problem. We show that ATL$_{ir}$ model checking is also $\mathbf{\Sigma_2^P}$-complete when the number of agents is a parameter. To demonstrate that the problem is $\mathbf{\Sigma_2^P}$-hard, we point out that:

Lemma 27 ATL *is semantically subsumed by* ATL$_{ir}$.

Proof. In order to transform a concurrent game structure M to a corresponding imperfect information concurrent game structure M', we fix the indistinguishability relations

function $eval_4(M, \varphi, choice)$**;**
Returns the states in which φ holds, given choices for all the coalitions from φ.

case $\varphi \in \Pi$ **:** return $\{q \mid \varphi \in \pi(q)\}$;
case $\varphi = \neg\psi$ **:** return $Q \setminus eval_4(M, \psi, choice)$;
case $\varphi = \psi_1 \vee \psi_2$ **:** return $eval_4(M, \psi_1, choice) \cup eval_4(M, \psi_2, choice)$;
case $\varphi = \langle\langle A\rangle\rangle T\psi$, where $T = \bigcirc$ or $\Box$ **:**
  $Q_1 := eval_4(M, \psi, choice|_2)$; $M' := trim(M, choice[1])$;
  add to M' new proposition $\overline{\mathsf{p}}$ with $\pi(\overline{\mathsf{p}}) = Q_1$;
  $Q_2 := mcheck_{CTL}(M', \mathsf{A}T\,\overline{\mathsf{p}})$;
  return $\{q \in St \mid \forall a, q' \,.\, a \in A \wedge q \sim_a q' \Rightarrow q' \in Q_2\}$;
case $\varphi = \langle\langle A\rangle\rangle \psi_1 \,\mathcal{U}\, \psi_2$ **:**
  c' := the number of cooperation modalities in ψ_1;
  $Q_1 := eval_4(M, \psi_1, choice|_2)$; $Q_2 := eval_4(M, \psi_2, choice|_{c'+2})$;
  $M' := trim(M, choice[1])$;
  add to M' new propositions $\overline{\mathsf{p}}_1, \overline{\mathsf{p}}_2$ with $\pi(\overline{\mathsf{p}}_1) = Q_1$, $\pi(\overline{\mathsf{p}}_2) = Q_2$;
  $Q_3 := mcheck_{CTL}(M', \mathsf{A}\overline{\mathsf{p}}_1 \,\mathcal{U}\, \overline{\mathsf{p}}_2)$;
  return $\{q \in St \mid \forall a, q' \,.\, a \in A \wedge q \sim_a q' \Rightarrow q' \in Q_3\}$;
end case

function $trim(M, thischoice)$**;**
Returns the CTL model, which includes exactly the transitions that can occur when A execute choices from $thischoice$.

- $\mathcal{R} := \emptyset$;   `// the CTL transition relation (contains pairs of states)`
- for each $q \in St$ and tuple $resp$ of choices from $\mathbb{A}\mathrm{gt} \setminus A$, such that $resp[a] \in d(a, q)$:
  - $q' := o(q, thischoice[q], resp)$;
  - $\mathcal{R} := \mathcal{R} \cup \{\langle q, q'\rangle\}$;
- return $\langle St, \mathcal{R}, \Pi, \pi\rangle$;

Figure 11: Nondeterministic algorithm for model checking formulae of ATL$_{ir}$; part II.

as the minimal total reflexive relations, (i.e. $\sim_a = \{\langle q, q\rangle \mid q \in St\}$ for all $a \in \mathbb{A}\mathrm{gt}$), which means that the agents can distinguish between any two states. Let φ be a formula of ATL, and φ' the result of adding subscript ir in every cooperation modality in φ. Then, $M, q \models \varphi$ iff $M', q \models \varphi'$. Thus, ATL model checking can be seen as a special case of ATL$_{ir}$ model checking. ■

function $mcheck_5(M, \varphi)$**;**
Returns the set of states in M, in which formula φ holds.

- assign cooperation modalities in φ with subsequent numbers $1, ..., c$;
  `// note that` $c \le l$
  `// we denote the coalition from the` i`th coop. modality in` φ `as` $\varphi[i]$
- for every $i = 1, ..., c$, assign the agents in $\varphi[i]$ with numbers $1, ..., k_c$;
  `// note that` $k_c \le k$ `and` $k_c \le l$
  `// we will denote the` j`th agent in` A `with` $A[j]$
- guess an array $choice$ such that, for every $i = 1, ..., c$, $q \in St$, and $j = 1, ..., k_c$, we have that $choice[i][q][j] \in d_{\varphi[i][j]}(q)$, and for every $q' \in St$ such that $q \sim_{\varphi[i][j]} q'$ we have $choice[i][q][j] = choice[i][q'][j]$;
  `// now, the optimal choices for all coalitions in` φ `are guessed`
  `// note that the size of` $choice$ `is` $\mathbf{O}(nkl)$
  `// by` $choice|_i$`, we denote array` $choice$ `with rows` $1, ..., i-1$ `removed`
- return $eval_5(M, \varphi, choice)$;

function $eval_5(M, \varphi, choice)$**;**
Returns the states in which φ holds, given choices for all the coalitions from φ.

case $\varphi \in \Pi$ **:** return $\{q \mid \varphi \in \pi(q)\}$;
case $\varphi = \neg\psi$ **:** return $Q \setminus eval_5(M, \psi, choice)$;
case $\varphi = \psi_1 \vee \psi_2$ **:** return $eval_5(M, \psi_1, choice) \cup eval_5(M, \psi_2, choice)$;
case $\varphi = \langle\langle A\rangle\rangle \bigcirc \psi$ **:**
  $Q_1 := pre_5(A, eval_5(M, \psi, choice|_2), M, choice[1])$;
  return $\{q \in St \mid \forall a, q' \,.\, a \in A \wedge q \sim_a q' \Rightarrow q' \in Q_1\}$;
case $\varphi = \langle\langle A\rangle\rangle \Box \psi$ **:** $Q_1 := St$; $Q_2 := Q_3 := eval_5(M, \psi, choice|_2)$;
  while $Q_1 \not\subseteq Q_2$ **do** $Q_1 := Q_1 \cap Q_2$; $Q_2 := pre_5(A, Q_1, M, choice[1]) \cap Q_3$
od;
  return $\{q \in St \mid \forall a, q' \,.\, a \in A \wedge q \sim_a q' \Rightarrow q' \in Q_1\}$;
case $\varphi = \langle\langle A\rangle\rangle \psi_1 \,\mathcal{U}\, \psi_2$ **:** c' := the number of cooperation modalities in ψ_1;
  $Q_1 := \emptyset$; $Q_2 := eval_5(M, \psi_1, choice|_2)$; $Q_3 := eval_5(M, \psi_2, choice|_{c'+2})$;
  while $Q_3 \not\subseteq Q_1$ **do** $Q_1 := Q_1 \cup Q_3$; $Q_3 := pre_5(A, Q_1, M, choice[1]) \cap Q_2$
od;
  return $\{q \in St \mid \forall a, q' \,.\, a \in A \wedge q \sim_a q' \Rightarrow q' \in Q_1\}$;
end case

Figure 12: The model checking algorithm refined (main part).

To show that the problem is $\mathbf{\Sigma_2^P}$-easy, we present a refinement of the algorithm from Section 6.2 in Figures 12 and 13.

function $pre_5(A, Q_1, M, thischoice)$**;**
Returns the set of states, for which the A's choices from $thischoice$ enforce that the next state is in Q_1, regardless of what agents from $\mathbb{A}\mathrm{gt} \setminus A$ do.

- $Q_2 := \varnothing$;
- for each $q \in St$: **if** $oracle_5(A, Q_1, M, thischoice, q) = yes$ **then** $Q_2 := Q_2 \cup \{q\}$ **fi**;
- return Q_2;

function $oracle_5(A, Q_1, M, thischoice, q)$**;**
Returns yes if, and only if, the A's choices from $thischoice$ in q enforce that the next state is in Q_1, regardless of what agents from $\mathbb{A}\mathrm{gt} \setminus A$ do.

- guess an array $resp$ such that, for every $a \in \mathbb{A}\mathrm{gt} \setminus A$, we have $resp[a] \in d(a, q)$;
  `// the most dangerous response from the opposition is guessed`
  `// note that the size of` $resp$ `is` $\mathbf{O}(k)$
- **if** $o(q, thischoice[q], resp) \in Q_1$ **then** return yes **else** return no **fi**;

Figure 13: The model checking algorithm refined: pre-image and oracle.

Proposition 28 *Function* $mcheck_5$ *defines a nondeterministic Turing machine that runs in time* $\mathbf{O}(n^2kl)$*, making calls to an* $\mathbf{NP}$ *oracle. The oracle itself is a nondeterministic Turing machine that runs in time* $\mathbf{O}(n + k)$*. The size of witnesses is never more than* $\mathbf{O}(nkl)$*.*

Proof. The main idea is as follows. First, we guess nondeterministically *all* the strategies for the cooperation modalities that occur in formula φ (we do it beforehand, as in Section 6.2). The strategies must be uniform, so setting $s_a(q)$ fixes automatically $s_a(q')$ for all $q \sim_a q'$. Then we model check φ recursively: for every subformula $\langle\langle A\rangle\rangle_{ir}\psi$, we assume the respective strategy and check the formula $\langle\langle\varnothing\rangle\rangle_{ir}\psi$. To do so, we take ATL formula $\langle\langle A\rangle\rangle\psi$ as input, and employ the standard ATL model checking algorithm from [5] with one important modification: every time function $pre(A, Q_1)$ is called, it assumes the respective A's choices, and checks whether $q \in pre(A, Q_1)$ by calling an $\mathbf{NP}$ oracle (*"is there a response from the opposition in* q *that leads to a state outside* Q_1*?"*) and reversing its answer. Note that the latter amounts to checking $M', q \models \langle\langle\varnothing\rangle\rangle\bigcirc\mathbf{Q_1}$, where M' is model M with A's actions fixed accordingly, and $\mathbf{Q_1}$ is a new proposition that holds exactly in states Q_1. Finally, we get rid of the states that have indistinguishable counterparts for which the assumed strategy is not successful. Note that, in the middle part of the algorithm, we use an adaptation of the ATL model checking procedure, which *iterates* over states of the system. This kind of iterative solution is possible because $\langle\langle\varnothing\rangle\rangle_{ir}\psi \equiv \langle\langle\varnothing\rangle\rangle\psi$ (although, of course, the analogous property does not hold for $\langle\langle A\rangle\rangle_{ir}$ in general).

The detailed algorithm is shown in Figures 12 and 13. The procedure is very similar to the ATL model checking algorithm from Section 3, which was used to demonstrate that the problem was $\mathbf{\Sigma_2^P}$-easy for ATL. Analogous complexity analysis applies: first, the number of iterations *within* one single call of function $eval$, as well as the number of calls to pre, is $\mathbf{O}(n)$; next, function pre runs in $\mathbf{O}(n)$ steps, including calls to the oracle; removing the states for which a member of the coalition can have any doubts can be done in time $\mathbf{O}(n^2k)$; finally, $eval$ is called at most $\mathbf{O}(l)$ times. In consequence, we get a nondeterministic polynomial algorithm that makes calls to an $\mathbf{NP}$ oracle. ■

Theorem 29 *Model checking* ATL$_{ir}$ *formulae over* i-CGS *is* $\mathbf{\Sigma_2^P}$*-complete.*

6.4 Discussion

The result has been somewhat surprising to us, since it turns out that a *fine grained* analysis puts checking strategic abilities of agents under imperfect information in the same complexity class as for perfect information games—while the first case appears *strictly* harder than the latter when we approach it from a more "distant" perspective (i.e. when the input parameters are less detailed). Let us recall from Section 3 that the hardness of model checking ATL is due to simultaneous actions of agents, and can be demonstrated even for scenarios that consist of a single step. It turns out that restricting agents' strategies to uniform strategies only does not increase model checking complexity *enough* to shift it to a higher complexity class. Even the size of witnesses is the same in both cases.

What is different then, that makes model checking of ATL$_{ir}$ *harder than* ATL *in relation to the number of transitions?*

Definitely *not* the number of transitions itself, because CGS can be seen as a special case of i-CGS. Comparison of model checking complexity for turn-based structures[7] can give us a hint in this respect. Note that, for such structures, $m = \mathbf{O}(nd)$ and we can use the model checking algorithms from Section 6.2 and from [5] to model-check formulae of ATL$_{ir}$ and ATL, respectively.

Proposition 30 *Model checking* ATL$_{ir}$ *over turn-based* i-CGS *is* $\mathbf{NP}$*-complete, while model checking* ATL *over turn-based* CGS *can be done deterministically in time* $\mathbf{O}(ndl)$*. Since* $d \leq n$ *for turn-based structures, the latter bound can be replaced by* $\mathbf{O}(n^2l)$*.*

The result can be generalised to systems in which only a fixed (or bounded) number of agents is acting in each state; we propose to call such systems *semi-turn-based* con-

[7] I.e., structures in which at every state there is a single agent who decides upon the next transition; this can be modelled by requiring that $d(a, q)$ is a singleton for all but one agent.

current game structures. Note that systems with a fixed (or bounded) number of agents are a special case of semi-turn-based CGS.

Proposition 31 *Model checking* ATL$_{ir}$ *over semi-turn-based* i-CGS *is* $\mathbf{NP}$*-complete, while model checking* ATL *over semi-turn-based* CGS *can be done deterministically in time* $\mathbf{O}(n^2 l)$.

Moreover, the exhaustive model checking of ATL formulae can be done in time $\mathbf{O}(nd^k l)$, while, for ATL$_{ir}$ formulae, it can be done in $\mathbf{O}(nd^{kn} l)$ steps. This is due to the fact that $\langle\langle A\rangle\rangle\Box\varphi \equiv \varphi \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Box\varphi$ and $\langle\langle A\rangle\rangle\varphi\,\mathcal{U}\,\psi \equiv \psi \vee \varphi \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\varphi\,\mathcal{U}\,\varphi$ in ATL, whereas analogous fixpoint characterisations do not hold for ATL$_{ir}$ modalities. Thus, successful ATL strategies can be computed incrementally, state by state. By contrast, uniform strategies must be considered *as a whole*, which requires much more backtracking if we check the possibilities exhaustively.

Nevertheless, we believe that the results in this section indicate that agent logics with incomplete information might not be unfeasible. If ATL formulae can be feasibly model-checked then agents with incomplete information are not *that* far away. And there already exist running model-checkers for ATL [6, 1], based on OBDD (Ordered Binary Decision Diagrams). Also, new model checking techniques, based on the idea of *Unbounded Model Checking*, are under development [24].

7 Conclusions

In this article, we established the precise model checking complexity for several variants of alternating-time temporal logic ATL. We analyzed the complexity of model checking for explicit models when the size of models is defined in terms of states rather than transitions, and the number of agents is considered a parameter of the problem. Most importantly, we proved that the problem is intractable for all studied variants of the logic. First of all, we showed that model checking ATL (with perfect information) over concurrent game structures is $\mathbf{\Sigma_2^P}$-complete. Moreover, for the previous semantics based on alternating transition systems, the problem is "only" $\mathbf{NP}$-complete, which suggests that using ATS may have some advantage over CGS. We also showed that ATL model checking over the broader class of *nondeterministic* ATS is still $\mathbf{NP}$-complete, and hence the ATS-based semantics might perhaps be used in a much more convenient way than until now. Finally, we proved that

1. model checking ATL$_{ir}$ (ATL with imperfect information) is $\mathbf{NP}$-complete in the number of transitions and the length of the formula (closing a gap in existing research), and

2. model checking ATL$_{ir}$ is $\mathbf{\Sigma_2^P}$-complete when the size of models is defined in terms of states rather than transitions.

	m, l	n, k, l	n_{local}, k, l
CTL	P [8]	P [8]	PSPACE [25]
ATL (ATS)	P [4]	**NP** (Section 4)	?
ATL (NATS)	P (Section 4.4)	**NP** (Section 4.4)	?
ATL (CGS)	P [5]	Σ_2^P (Section 3)	?
ATL_{ir}	NP ([34] + Section 6.2)	Σ_2^P (Section 6.3)	?

Figure 14: Model checking complexity: completeness results for various settings of input parameters. Symbols n, k, m stand for the number od states, agents and transitions in the explicit model, l is the length of the formula, and n_{local} is the number of local states in a concurrent program.

3. Thus, checking strategic ability under imperfect information falls in the same complexity class as checking strategic ability for *perfect* information agents, when a more refined analysis is conducted – which we consider somewhat surprising.

We summarise the existing results on model checking ATL-related logics in Figure 14.

Additionally, a truth-preserving translation of ATL models and formulae is presented. The resulting models are always *turn-based*, which usually means an exponential decrease in the number of transitions. As turn-based alternating transition systems are very close to CTL models, as well as extensive form games with perfect information, one may hope that some interesting techniques can be transferred from CTL model checking and/or game theory this way. Moreover, the translation of models is independent from the translation of formulae in our construction, which allows for "pre-compiling" models when one wants to check various properties of a particular multi-agent system.

We would like to thank: Thomas Eiter for his help in the **NP**-hardness proof of the *sfc-SAT* problem; Rafał Somla for the discussions about model checking ATL and its various extensions; Nils Bulling for checking our checking of "weak until"; Pierre-Yves Schobbens for comments; last, but not least, the anonymous reviewers of MFCS'05, CEEMAS'05, ICTCS'05 and BNAIC'05 for their helpful remarks.

References

[1] R. Alur, L. de Alfaro, R. Grossu, T.A. Henzinger, M. Kang, C.M. Kirsch, R. Majumdar, F.Y.C. Mang, and B.-Y. Wang. jMocha: A model-checking tool that exploits design structure. In *Proceedings of ICSE*, pages 835–836. IEEE Computer Society Press, 2001.

[2] R. Alur and T. A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

[3] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 100–109. IEEE Computer Society Press, 1997.

[4] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Lecture Notes in Computer Science*, 1536:23–60, 1998.

[5] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.

[6] R. Alur, T.A. Henzinger, F.Y.C. Mang, S. Qadeer, S.K. Rajamani, and S. Tasiran. MOCHA user manual. In *Proceedings of CAV'98*, volume 1427 of *Lecture Notes in Computer Science*, pages 521–525, 1998.

[7] A. L. Blum and M. L. Furst. Fast planning through graph analysis. *Artificial Intelligence*, 90:281–300, 1997.

[8] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[9] L. de Alfaro, T.A. Henzinger, and F.Y.C. Mang. The control of synchronous systems. In *Proceedings of CONCUR 2000*, volume 1877 of *Lecture Notes in Computer Science*, pages 458–473, 2000.

[10] L. de Alfaro, T.A. Henzinger, and F.Y.C. Mang. The control of synchronous systems, part II. In *Proceedings of CONCUR 2001*, volume 2154 of *Lecture Notes in Computer Science*, pages 566–580, 2001.

[11] T. Eiter. Oral communication. March 2005.

[12] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier Science Publishers, 1990.

[13] E.A. Emerson and J.Y. Halpern. "sometimes" and "not never" revisited: On branching versus linear time temporal logic. In *Proceedings of the Annual ACM Symposium on Principles of Programming Languages*, pages 151–178, 1982.

[14] E.A. Emerson and J.Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30(1):1–24, 1985.

[15] V. Goranko. Coalition games and alternating temporal logics. In J. van Benthem, editor, *Proceedings of TARK VIII*, pages 259–272. Morgan Kaufmann, 2001.

[16] V. Goranko and W. Jamroga. Comparing semantics of logics for multi-agent systems. *Synthese*, 139(2):241–280, 2004.

[17] W. Jamroga. Some remarks on alternating temporal epistemic logic. In B. Dunin-Keplicz and R. Verbrugge, editors, *Proceedings of Formal Approaches to Multi-Agent Systems (FAMAS 2003)*, pages 133–140, 2003.

[18] W. Jamroga. *Using Multiple Models of Reality. On Agents who Know how to Play Safer*. PhD thesis, University of Twente, 2004.

[19] W. Jamroga and J. Dix. Do agents make model checking explode (computationally)? In M. Pěchouček, P. Petta, and L.Z. Varga, editors, *Proceedings of CEEMAS 2005*, volume 3690 of *Lecture Notes in Computer Science*, pages 398–407. Springer Verlag, 2005.

[20] W. Jamroga and J. Dix. Model checking strategic abilities of agents under incomplete information. In M. Coppo, E. Lodi, and G.M. Pinna, editors, *Proceedings of ICTCS 2005*, volume 3701 of *Lecture Notes in Computer Science*, pages 295–308. Springer Verlag, 2005.

[21] W. Jamroga and J. Dix. Turning game models turn-based for model checking properties of agents. In Katja Verbeeck, Karl Tuyls, Ann Nowé, Bernard Manderick, and Bart Kuijpers, editors, *Proccedings of BNAIC*, pages 143–150, 2005.

[22] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2–3):185–219, 2004.

[23] G. Jonker. Feasible strategies in Alternating-time Temporal Epistemic Logic. Master thesis, University of Utrecht, 2003.

[24] M. Kacprzak and W. Penczek. Unbounded model checking for Alternating-time Temporal Logic. In *Proceedings of AAMAS-04*, 2004.

[25] O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

[26] F. Laroussinie. About the expressive power of CTL combinators. *Information Processing Letters*, 54(6):343–345, 1995.

[27] F. Laroussinie, N. Markey, and Ph. Schnoebelen. Model checking CTl^{+} and FCTL is hard. In *Proceedings of FoSSaCS'01*, volume 2030 of *Lecture Notes in Computer Science*, pages 318–331. Springer, 2001.

[28] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, 1995.

[29] K.L. McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem*. Kluwer Academic Publishers, 1993.

[30] R.C. Moore. A formal theory of knowledge and action. In J. Hobbs and R.C. Moore, editors, *Formal Theories of the Commonsense World*. Ablex Publishing Corp., 1985.

[31] L. Morgenstern. Knowledge and the frame problem. *International Journal of Expert Systems*, 3(4), 1991.

[32] C.H. Papadimitriou. *Computational Complexity*. Addison Wesley : Reading, 1994.

[33] W. Quine. Quantifiers and propositional attitudes. *Journal of Philosophy*, 53:177–187, 1956.

[34] P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2), 2004.

[35] W. van der Hoek. Formal comment on W. Jamroga's paper. Presented at FAMAS'03, 2003.

[36] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In C. Castelfranchi and W.L. Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, pages 1167–1174. ACM Press, New York, 2002.

[37] W. van der Hoek and M. Wooldridge. Cooperation, knowledge and time: Alternating-time Temporal Epistemic Logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[38] S. van Otterloo and G. Jonker. On Epistemic Temporal Strategic Logic. In *Proceedings of LCMAS*, pages 35–45, 2004.

[39] M. Wooldridge. *Reasoning about Rational Agents*. MIT Press : Cambridge, Mass, 2000.